Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 12-023

Information lifetime aware analysis for dynamic social networks

Venkata M.V. Gunturi, Kenneth Joseph, Shashi Shekhar, Kathleen M. Carley

October 25, 2012

A

Information lifetime aware analysis for dynamic social networks

Venkata M.V. Gunturi, Dept of Computer Science and Engineering, University of Minnesota, USA
Kenneth Joseph, Institute for Software Research, School of Computer Science, Carnegie Mellon University, USA
Shashi Shekhar, Dept of Computer Science and Engineering, University of Minnesota, USA
Kathleen M. Carley, Institute for Software Research, School of Computer Science, Carnegie Mellon University, USA

Given a dynamic social network, a discrete time interval, and a bound on the lifetime of a unit of information, the Dynamic INFormation Liaison (DINFL) problem aims to find a set of individuals that includes the most "information relaying" (or central) person for each time in the given interval. This problem is important for several societal applications related to viral marketing, word-of-mouth marketing, finding key influences etc. However, DINFL poses both semantic and computational challenges. Semantic challenges requires us to model realistic pathways through which information can flow. This involves modeling semantics of information flow such as bounded lifetime of information (i.e. discussions on a topic last only finite amount of time). On the other hand, computational challenges arise due the inherent non-stationarity within a dynamic social network. Consequently, centrality of individuals in the network change with time. Related work in this area has been limited due to two reasons. First, existing computational methods to solve DINFL are likely to incur exorbitant computational costs as they would have to compute the centrality value of each time instant in the given time interval to ensure correctness. Second, they lack adequate models to capture the semantics of information flow, such lifetime of information. In contrast, this paper proposes a novel centrality metric called, *Information lifetime aware (or InLife)* betweenness centrality, which considers the semantic issues of information flow. A detailed case study shows how our metric compares with the traditional notion of betweenness centrality. We also propose novel *flow epoch* computational technique that allows us to prune certain time intervals where centrality would not change. This helps us to compute the value of metric over large *dynamic social networks* spread over long periods of time. We prove our computational technique is correct and complete and provide experimental results showing it outperforms an alternative method by orders of magnitude.

General Terms: Social Events,Dynamic Network Measures,Betweenness Centrality

1. INTRODUCTION

Given a dynamic social network, a discrete time interval, and a bound on the lifetime of a unit of information, the Dynamic INFormation Liaison (DINFL) problem aims to determine a set of individuals who are serving as the primary liaisons for relaying information across the network and their corresponding time instances. This paper refers to such people as being central. For instance, consider the dynamic social network shown in Figure 1. Here, Mary can be considered as liaison at 9:00am as she forwards the information about GO1 and GO2 (received from John) to Sara in a timely fashion. However, the third offer (GO3) sent by John is not forwarded by Mary in timely fashion. This could be due to the fact that Mary does not check her email between 12:00noon and 5:00pm. However, Jack, who checks emails in the afternoon (as a example) forwards the offer to Sara in time. Thus, a possible output of our problem for this input would be: (a) Mary is central (primary liaison) for 9:00am and 10:00am, (b) Jack is central for 12:00noon. The primary problem considered in this paper is development of computationally scalable techniques to find such central individuals in large dynamic social networks.

Finding information liaisons and their respective time instants (e.g. Mary was central for only 9:00am and 10:00am in our previous example) is important for societal applications like viral marketing, word-of-mouth marketing, key influencers etc. It could be argued that that it is enough to know the central individuals for a particular

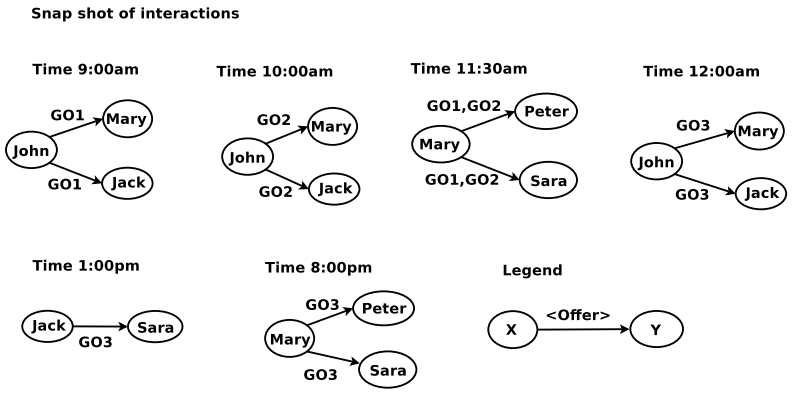

Fig. 1. Sample social event data. Here, John, a manager at viral marketing company such as Groupon, has to market three offers GO1, GO2, GO3. Where, he receives GO1, GO2, GO3 at times 9:00am, 10:00am and 12:00noon respectively and each one of them expiries within 2hrs.

time interval, and not for the corresponding time instants within the time interval. By such an approach, Mary and Jack (Figure 1) would be considered as information liaisons for the entire time period 9:00am through 12:00noon. This may not be suitable for following reasons. First, by not targeting the specific liaisons, one may have to send out a large number of emails creating issues of Spam and nuisance. Second, in many kinds of media there is usually an associated cost of posting an advertisement. For example, consider the scenario of marketing through online social networks such as Facebook, or blogs. Here, there is an associated cost of posting an advertisement on one's facbook profile or blog.

A key step for determining liaisons (or central) nodes in dynamic social network is to compute the number of the candidate paths passing through any particular node. These paths, referred to as *information pathways*, represent potential conduits for information flow. For instance, the path created out of events John sending an email to Mary at 9:00am and Mary sending an email to Sara at 11:30am is a candidate information pathway. Understandably then, individuals lying on a large number of these pathways can be considered as more central. However, determining these information pathways pose both semantic and computational challenges. Semantic challenges require us to consider only those candidate paths along which the same piece of information might feasibly move. This requires us to consider the following two properties of candidate information pathways. First, the pathways should reflect an appropriate frame of reference for information flow. In other words, the social events constituting an information pathway should be *temporally ordered* [Nia et al. 2010; Howison et al. 2011; Kossinets et al. 2008; Tang et al. 2010; Lerman et al. 2010]. For instance, a pathway created out of events, John sending an email to Mary at 12:00noon and Mary sending an email to Sara at 11:30am, is not a valid path. As we show later consideration of these kind of paths is a major limitation of aggregation based techniques.

Second, since discussion of a particular topic (or equivalently, information) does not last forever [Wu and Huberman 2007; Weng et al. 2012; Tyler and Tang 2003; Karagiannis and Vojnovic 2008], it is vital to find some plausible way to bound the "lifetime" of a piece of information in a given network. Thus, candidate paths with long waits should not be considered. For instance, the path created out of the events (a) John sends an email to Mary at 9:00am and (b) Mary sends an email to Peter at 8:00pm

may not be a valid information path as information waits at Mary for 11 hours. This wait could either result in expiry of the offer (which is the case here) or it might be full (in case of first-come-first-serve offers). Either way, paths with such long waits are not usually desirable.

The computational challenge for determining the central individuals arises from the fact that information pathways change with time in a dynamic social network. We refer to this property as *non-stationarity*. For instance, the information pathway between Jack and Sara in Figure 1 at 9:00am passes through Mary, whereas, the same information path passes through Jack at 12:00noon. Consequently, the centrality of individuals needs to be re-computed to ensure correctness. This becomes a major performance bottleneck if the desired metric needs to be computed for several time instants.

In social network research, the traditional metric for identifying central nodes within a flow of information is based on the notion of betweenness centrality [Freeman 1977; Anthonisse 1971]. However, this notion assumes a static view of the social network. Although, there have been attempts [Tang et al. 2010; Habiba et al. 2007; Kim and Anderson 2012] to extend this to temporal cases, little work has been done which considers all the semantic and scalability challenges of the metrics. For instance, [Tang et al. 2010; Habiba et al. 2007; Kim and Anderson 2012] do model the information pathways as sequences of "temporally ordered" events. However, they do not consider the lifetime of the information at an individual node. Consequently, their approach might overestimate the centrality of agents[1] in some cases. For example, they would consider the path resulting from events (a) John sending an email to Mary at 9:00am and(b) Mary sending an email to Peter at 8:00pm as a valid information pathway despite a time lag of 11 hours. Moreover, their approach would encounter significant computational bottlenecks in case large datasets as they would have re-compute the value of centrality for each time instant to ensure correctness.

Contributions: In order to address the semantic challenges of our liaison identification problem, we propose a novel centrality metric called *Information Lifetime aware (InLife)* betweenness centrality. This centrality metric incorporates an appropriate frame of reference for information flow by considering only temporally-ordered information pathways. The metric also respects the lifetime of the information by bounding the wait at any intermediate node.

We develop a novel computational technique, called *flow epoch* based approach which helps to achieve scalability while computing InLife betweenness centrality on very large datasets. We also prove the correctness and completeness of our computational technique. Detailed experimental evaluation of the proposed techniques are performed on large real datasets. Additionally, we also provide a case study which addresses the following questions:

— How does InLife betweenness centrality compare with the traditional betweenness centrality calculated on the network obtained by aggregating the events into panels?
— Does an individual's circadian rhythms play a role in his/her InLife betweenness centrality?

Illustrative application domain: Information diffusion The study of Information diffusion primarily concerns the following two aspects with the information flow on

[1] In this we paper we do not differentiate between the terms agent and individual and use them interchangeably.

the underlying social networks: (a) Understanding the very nature of the information flow through mathematical models [Yang and Leskovec 2010; Myers et al. 2012], (b) Tracking of ideas, memes, information units [Weng et al. 2012; Lerman and Ghosh 2010; Leskovec et al. 2009; Kleinberg 2002; Wu and Huberman 2007] in the network.

Besides realistic mathematical models, one of the key results of these studies, and the most applicable to our work, was the notion of the *lifetime* of an idea or meme. The *lifetime* of an idea, meme or an information unit is defined as the maximum number of consecutive time units for which there are posts about that idea or meme [Weng et al. 2012]. For instance [Wang et al. 2011] notes that the median delay between a person receiving an email and forwarding a message with the same subject was between 5 and 17 hours for different types of people within an organization. Implicitly, this tells us that information cascade on a topic or idea has a bounded lifetime in the network. This could be due to a variety of reasons including limited attention of individuals and the structure of the network [Wu and Huberman 2007; Weng et al. 2012].

Scope: In this work, we focus on cases where the dataset does not contain knowledge on the information flowing between two agents, and thus an assumption of the lifetime of information is appropriate and quite necessary for finding information liaisons. While the model given here is appropriate only under these conditions, we note that (a) such data is much more prevalent than data containing information on precisely what flows between two agents, b) such information can lead to questions of the privacy of individuals within the dataset. Nevertheless, the proposed *flow epoch* computational technique to solve DINFL can be easily applied to dataset containing contextual knowledge of the information flow (e.g. re-tweets etc) without compromising its correctness or completeness.

Outline: The rest of the paper is organized as follows. Section 2 introduces the *time aggregated graph*, which is the representational model used in this paper and gives a formal definition of *InLife Betweenness* centrality. We give detailed description of related work talk about its limitations in Section 3. The proposed *flow epoch* based computational technique is detailed in Section 4. Section 5 describes a *flow epoch* based algorithm to compute *InLife* betweenness centrality. We give a formal correctness and completeness proof of our approach in Section 6. Detailed experimental analysis of the proposed computational technique is given in Section 7. Finally, we give a case study in Section 8 and conclude with a hint of future directions.

2. BASIC CONCEPTS AND PROBLEM DEFINITION

A *dynamic social network*, can be defined as any collection of social event that can be mapped to a specific instance in time. One form of event data of particular interest are relational events, which can be described by the tuple $< A_i, A_j, T >$, where A_i and A_j are two people (or agents), and T is a timestamp. Each relational event creates a temporary link between A_i and A_j, which can then be used to create networks in a variety of ways. Relational events can represent several types of social interactions such as emails, phone calls, face-to-face interaction, co-location pairs etc. In this paper we model social event data through email interactions.

Figure 2 illustrates a sample dynamic social network represented as a series of snapshots at different times. A particular snapshot represents all the events occurring at that time instant. For example, the top left snapshot (taken at time $t = 1$) in Figure 2 constitutes the following three relational events: (1) E sending an email to A; (2) A sending an email to B and; (3) C sending an email to D.

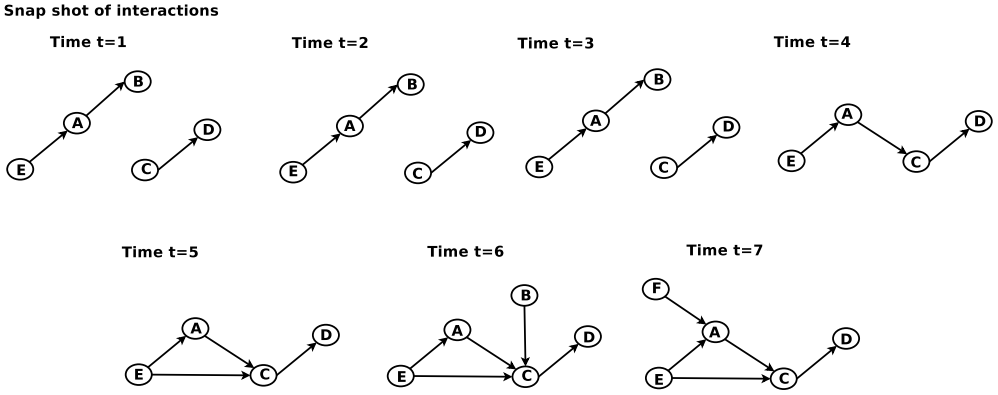

Fig. 2. Sample dynamic social network

Given such dynamic social network data, a key task while computing *InLife* betweenness centrality would be to retrieve candidate information paths. Semantically these paths are usually modeled as a set of "temporally-ordered" interactions [Habiba et al. 2007; Nia et al. 2010; Kossinets et al. 2008] across time instants. Now, the method of retrieving the information paths depends on the choice of the model for the data. Over the past few years several models have been proposed [Habiba et al. 2007; Tang et al. 2010; Kim and Anderson 2012; Tantipathananandh et al. 2007] for modeling dynamic social networks. We broadly classify these models into: (1) Snapshot models, and (2) Explicit Info-path models. We now briefly describe these models and talk about their applicability for our problem.

2.1. Representational models

Snapshot model: This model represents the dynamic social event data as a series of snapshots taken at different time instants [Tang et al. 2010; Kim et al. 2012]. Figure 2 uses this model to represent the data. Now, a possible information chain, $E \rightarrow A \rightarrow B$ (starting at E at $t = 1$) can be obtained by combining the edge (E, A), (from snapshot at $t = 1$), with the edge (A, B) (from snapshot at $t = 2$), assuming it takes one time unit for the information to flow in the edge (E, A).

Explicit info-path models: Another popular model found in the literature [Kim and Anderson 2012; Habiba et al. 2007] is based on the idea of explicitly representing the "temporally-ordered" information paths in a graph. Figure 3 illustrates one such approach with the social event data shown in Figure 2. Here, the node corresponding to every agent is replicated across the time instants and directed edges are added in a "temporally-ordered" sense. For instance, consider the event when E sends an email to A at time $t = 1$ (see Figure 2). This interaction is represented by adding a directed edge between the copy of node E at $t = 1$ and A at $t = 2$ [2]. Now, the path $E \rightarrow A \rightarrow B$ (starting at E at $t = 1$) can be viewed as a simple directed path between node E at t=1, A at t=2, B at t=3 (shown in bold in Figure 3).

While these models offer certain advantages, in many cases they could either be inconvenient or too restrictive. For instance, Explicit info-path models may not be able

[2]Again we assume that it takes one time unit for the information to flow in the edge (E, A)

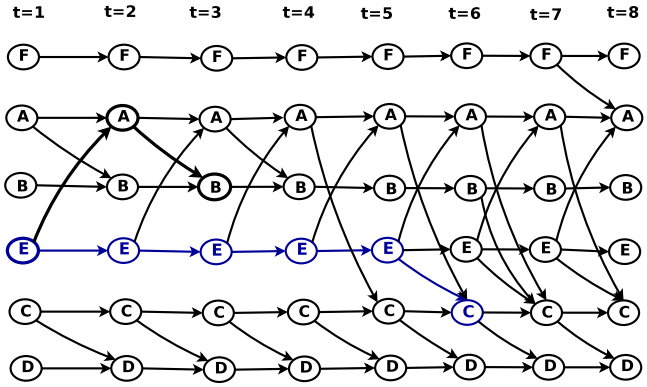

Fig. 3. Explicit representation of information paths

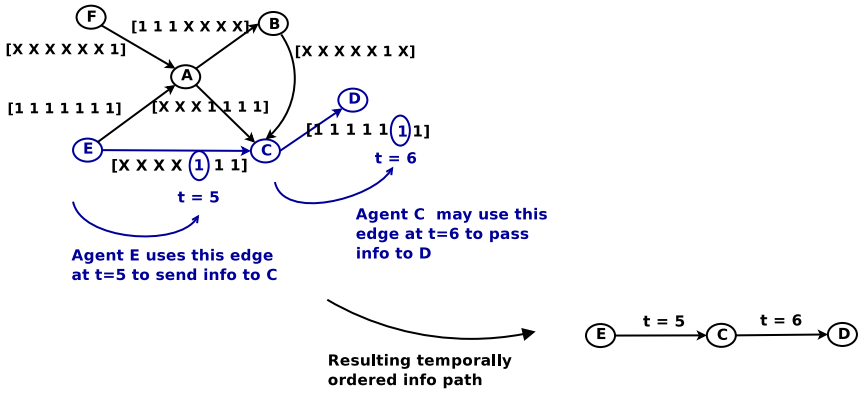

Fig. 4. Time aggregated graph

to incorporate the notion of the lifetime of information at a node. In other words these methods assume that information may live forever at a node, resulting in unrealistic information chains. For example, consider the blue colored information path in Figure 3. Here, the information needs to wait at node E for 4 time steps, which may not be desirable in certain cases. Moreover, this model would replicate the node information across the time instants, which may not efficient in the case of large datasets. On the other hand, a snapshot model may be inconvenient as it would require several switches across the snapshots in order to retrieve the desired paths. In contrast, we propose to use a more compact model based *time-aggregated graph* [George et al. 2007; George and Shekhar 2006]. Not only does this representation model the information paths easily, but it also reduces the replication across time instants.

Time aggregated graph: This model assigns a time series, called an *interaction time series*, to each edge. This time series stores the time instants at which interactions have occurred. Figure 4 illustrates this model with the network shown in Figure 2. Here, edge (A, B) is associated with an interaction time series $[1\ 1\ 1\ X\ X\ X\ X]$. This implies that A interacted with B at times $t = 1, 2, 3$. Using this model, the information paths

can be easily and efficiently retrieved from the *time-aggregated* graph by following the edges in a temporally-ordered fashion (see Figure 4).

2.2. Information lifetime aware betweenness centrality

The centrality of an individual in an information cascade can be captured by recording the number of information paths passing through that individual. Understandably then, individuals who lie on a large number of pathways through which information is likely to spread most efficiently are generally thought as central. In this paper we assume that information tends to flow through the shortest path possible from one person to another. Thus, as we discuss our metric, we will often refer to "information pathways", where we mean temporally ordered shortest paths between two agents.

Traditionally, betweenness centrality [Freeman 1979; 1977; Anthonisse 1971] was used to capture such a notion of centrality. This paper takes this definition and builds on the efforts of previous works [Tang et al. 2010; Habiba et al. 2007]. Specifically, we consider two improvements in connecting a betweenness metric to the traditional notion of betweenness centrality as centrality in information flow. First, by introducing the notion of *lifetime of information* at a node, we relax the assumption that all possible connections between two people are the result of one continuous piece of information flow. This operationalization is much more in line with notions of social capital [Coleman 1988] in that pathways which are recurrent are likely indicative of some stronger notion of social connection than a mere information flow. Second, like previous works in this area [Tang et al. 2010; Habiba et al. 2007], we ensure that all pathways used in generating the metric are temporally ordered, and thus that we only consider paths where information could realistically have moved from one agent to another. We now define formally define our proposed centrality metric, *Information lifetime aware betweenness centrality* (InLife betweenness centrality).

The *InLife betweenness centrality* of a node v at time t, $C_{B\alpha}(v)[t]$, is defined in Equation 1. Here, $\sigma_{sd}[t]$ refers to the number of shortest paths which start at time t between node s and node d, and $\sigma_{sd}(v)[t]$ refers to the number of shortest paths, starting at time t between node s and node d that pass through node v. Here, a shortest path, $P_{sd} = s, x_1, x_2, x_3 \ldots, d$, between s and d has two additional constraints. First, all the nodes in P_{sd} should be visited in a *temporally ordered* sense. This means that if x_i, x_j and x_k are three consecutive nodes in P_{sd} visited at times t_i, t_j and t_k respectively, then $t_i < t_j < t_k$. The second constraint on P_{sd} requires that maximum wait duration at any node is less than α. In our previous example with x_i, x_j and x_k, this constraint implies that $t_j - t_i \leq \alpha$ and $t_k - t_j \leq \alpha$. We refer to α as the *info-lifetime* parameter. From this point, as we use the phrase shortest path or Information pathway, the above mentioned definition is implied.

$$C_{B\alpha}(v)[t] \; = \sum_{\forall s \neq v \neq d \in V} \frac{\sigma_{sd}(v)[t]}{\sigma_{sd}[t]} \tag{1}$$

2.3. Problem Definition

Input: The input of the problem consists of (a) dynamic social network dataset, $S = e_1, e_2, \ldots, e_{|S|}$, consisting of time stamped interactions e_i among a set of agents V. Here, each interaction $e_i = (x, y, t)$, $x, y \in V$ refers to the event of x sending an email to y at time t; (b) a time interval λ represented as discrete time instants; (c) an *Info-lifetime* parameter.

Agent	Start-times
A	1,2,3
C	4,5,6

Fig. 5. Output of the problem

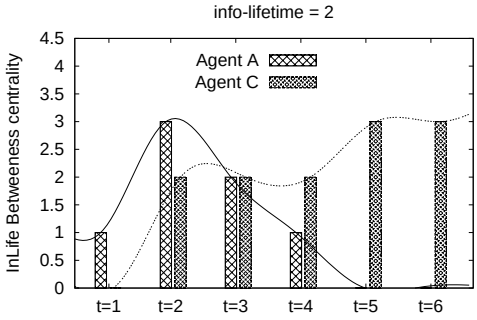

Fig. 6. InLife betweenness centrality values for data in Figure 2

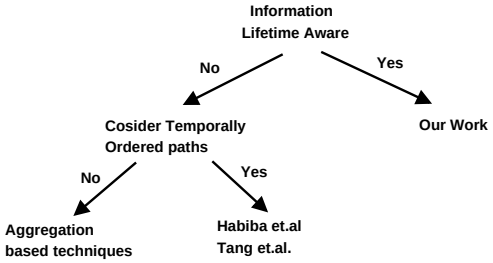

Fig. 7. Related work classification

Output: A set of agents γ, where each agent $v \in \gamma$ is associated with a set of time instants τ_v.

Objective: Each agent $v \in \gamma$ has highest InLife betweenness centrality (given by Equation 1) for times $t \in \tau_v$

Constraints: Analysis is limited to the time window λ under consideration i.e., interactions which have occurred outside λ are assumed to have no effect on the current problem instance.

Figure 5 and Figure 6 illustrate an instance of the problem. Here, the input consists of following: (a) the dynamic social network shown in Figure 2; (b) $\lambda = 1, 2, 3, 4, 5, 6$; (c) an *Info-lifetime* parameter of 2. As shown in Figure 5, agent A has higher centrality for times $t = 1, 2, 3$ (ties broken arbitrarily), whereas agent C is central at times $t = 4, 5, 6$. Figure 6 shows the individual values of *InLife* betweenness centrality for A and C at all time instants in λ. Note that centrality values for all other individuals in Figure 2 would be zero.

3. RELATED WORK AND ITS LIMITATIONS

Over past years there have been works attempting to address challenges associated with computing path based centrality metrics such as betweenness centrality on dynamic social networks. These can be broadly classified on the basis of whether the lifetime of information is considered or not (see Figure 7). Work done under the as-

sumption of unbounded lifetime of information can be further divided on the basis of consideration for temporally ordered paths. This division leads to two broad categories: (a) Aggregation based techniques [Börner et al. 2004; Börner et al. 2005; Barabsi et al. 2002; Diesner et al. 2012]; (b) Computation on raw event data [Habiba et al. 2007; Tang et al. 2010].

Aggregation based techniques create a set of network panels from the social event data, where each panel is made by aggregating the social events spread over a certain time duration (day, week, month etc). After aggregation, traditional static betweenness metrics are used on each panel. This method suffers from its requirement that the researcher determine some a priori notion of a time period in which all events are modeled as having occurred at the exact same time - that is, the research must select a "level of aggregation". Though there are increasingly useful automated techniques for doing so [Sulo et al. 2010], and certain time periods, such as a week, might make perfect sense for certain research questions, this aggregation may still cause three important issues to surface when analyzing results.

First, aggregation requires the presumption that changes to agent centrality occur at predetermined, discrete intervals. Consequently, agents who have only short bursts of activity within an interval might be considered high in centrality within an entire panel. As a contrived example, someone who leaves a company in the middle of a week might be deemed central for the entire week because each panel was aggregated for a week's duration. Second, aggregation loses the temporal information as to which paths within the network were created in what temporal order [Howison et al. 2011]. For instance, a network created by aggregating the snapshots shown in Figure 2 would have $F \rightarrow A \rightarrow B$, which is not a valid path as agent A cannot relay the information received from F (at time $t = 7$) to B, since interactions with B were at an earlier time. As a result metrics such as betweenness centrality computed over these panels would consider information paths which did not actually exist in the underlying event data [Nia et al. 2010; Howison et al. 2011].

Because of these issues, several attempts [Habiba et al. 2007; Tang et al. 2010; Kim and Anderson 2012] have been made to develop computational techniques which can be utilized on the raw social network data without the need to aggregate into panels. While they address the representational issue by considering only temporally ordered paths, they do not consider the lifetime of the information flow. Consequently, they may overestimate the number of information paths passing through a node by considering the interactions which are far ahead in this future. For instance, consider again the data shown in Figure 2. Here, agent C is active only at later times ($t = 4, 5, 6$). Now, if this is not taken into account, we would consider information paths which have long waits. Consequently, we would overestimate the centrality value. Figure 8(b) illustrates this effect for data in Figure 2. Here, we have over estimated the number of information paths passing through node C for time $t = 1, 2, 3$. Moreover, these approaches would incur significant computational overhead while analyzing large datasets containing social events spread over long periods of time (e.g. months or years).

4. PROPOSED COMPUTATIONAL APPROACH

One of the main challenges while computing InLife betweenness centrality over dynamic social networks is that centrality values of agents change with time. For instance, centrality of agent A in Figure 2 for times $t = 2, 3, 4, 5$ was $3, 2, 1$ and 0 respectively (see Figure 6). This happens because shortest paths between agents change with time. For instance, consider Figure 9 which shows the shortest path trees for node E (on data shown in Figure 2) for times $t = 1, 2, 3, 4, 5, 6$. We can observe that the shortest path between E and C changes at time $t = 5$ and similarly, that the shortest path to B does not exist after time $t = 2$. Such changes affect the InLife betweenness centrality

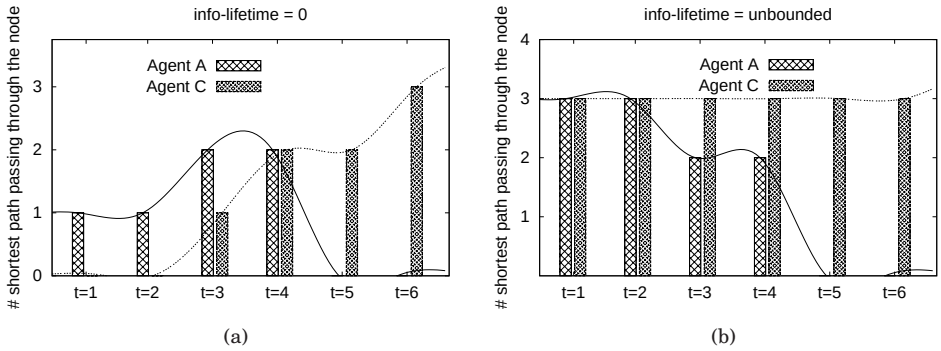

Fig. 8. Effect of lifetime on #shortest paths passing through an agent.

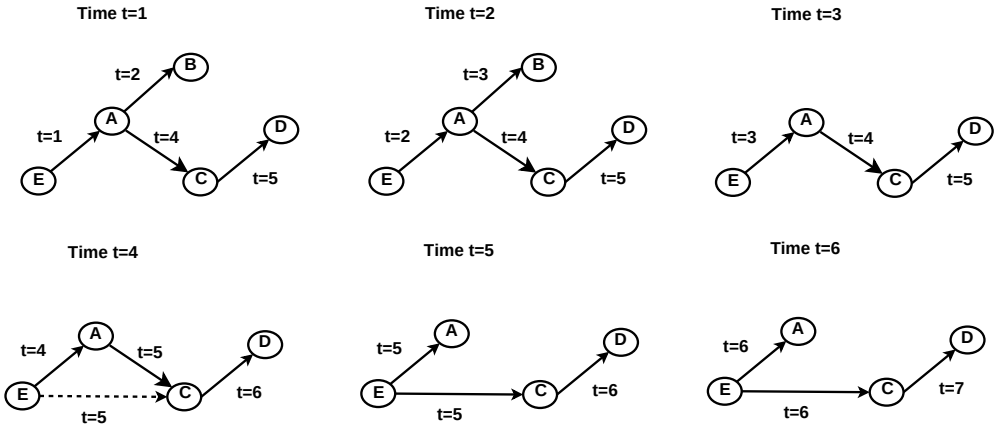

Fig. 9. Shortest path tree of node E for different times.

of agents. Specifically, the drop in the centrality of A is due to the fact that there are progressively fewer numbers of shortest paths passing through A as time progresses. Now, a naive approach to address this challenge would be to compute shortest paths for all the time instants and post-process the result to get the centrality value. While this approach may be suitable for analyzing the data a few time instants, it would incur exorbitant computational costs when one is interested in analyzing for longer periods of time.

This paper proposes a divide and conquer based approach to address this challenge. This approach divides the given time interval into a set of smaller disjoint time intervals. Inside these intervals the shortest path between the agents is guaranteed not to change, in other words the ranking of candidate paths is stationary. Now, we can employ a dynamic programming (DP) based approach within these intervals to compute the shortest paths. For instance, consider the case where the shortest paths from agent E to all the other nodes (see Figure 9) need to be computed for times $t = 3, 4, 5, 6$. Here, the shortest path tree changed at $t = 5$, which divides $[3\ 6]$ into two sub-intervals $[3\ 5)$ and $[5\ 6]$. We refer to $t = 5$ as a *flow epoch point*. Now, we can execute two instances of a DP based approach (one over each of the sub-intervals) to compute the shortest paths

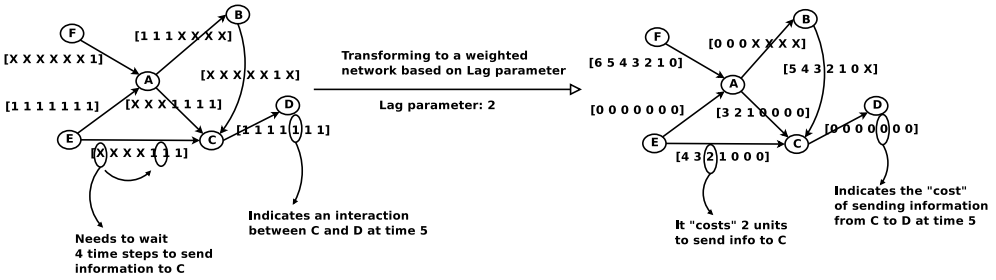

Fig. 10. "Cost" transformation: First step of Earliest relay Transformation

from agent E to all the other nodes. Notice that a naive approach would have needed four instances of DP based approach, whereas our approach needs only two. Before we delve into details of our approach to compute the *flow epoch points*, we first describe our approach of incorporating the *Info-lifetime* parameter into the algorithm.

4.1. Earliest relay transformation

In order to incorporate the *Info-lifetime* parameter, we propose a two step transformation of the original interaction series after which we can both model the bounded lifetime and also use efficient path counting techniques such as [Brandes 2001]. We refer to this transformation as *Earliest relay transformation* (ERT). The first step of this transformation involves associating a certain amount of "cost" for waiting. Internally, this is again a two step process. Here, we first we convert 1's in the time series to 0 and then assign a non zero cost to positions with X's. For instance, for the interaction time series of the edge (E,C) in Figure 10, the result of the first step on [X X X X 1 1 1] would be [X X X X 0 0 0]. Now, we would assign a non-zero cost to positions with X's. This can be done in several ways depending on the application in context. In this paper, we use a linear cost model, where the cost assigned is simply the duration (from current time) for which the agent needs to wait before sending information. Thus, the result of the second step on the above interaction time series would be [4 3 2 1 0 0 0]. The result of the first step ERT is shown in Figure 10. Intuitively, one can view the values in the interaction time series after this step as the duration for which one has to wait before relaying the information.

Now, the second step of *Earliest relay transformation* (ERT) involves converting the resulting time series from the previous step into a suitable form for use with a DP based algorithm. For instance, after the first step the interaction series would have 0's. These would need to be converted to represent the earliest time instants when the next edge could be taken. Figure 11 illustrates this conversion. Here, we add the value of (time index + 1) [3] to each entry in the resulting time series of the previous step. For example, our resulting time series from step one ([4 3 2 1 0 0 0]) would be converted to [6 6 6 6 6 7 8]. Now, the values in this time series can be interpreted in the following way: The time series has a value of 6 at $t = 1$. This means that we can access the next edge from the end node (node C in this case) at time $t = 6$. Another way to interpret this would be; we need to wait for 4 $(6 - 1 - 1)$ time units before accessing the next edge. The second interpretation makes it easier to model the bounded lifetime of information at a node. The threshold *Info-lifetime parameter* included in the input denotes this bounded lifetime. For example, a *Info-lifetime parameter* of 2 would imply that the information can wait for a maximum duration of 2 time units. Therefore, in

[3]We add an additional 1 because the information takes one time unit to travel on the edge

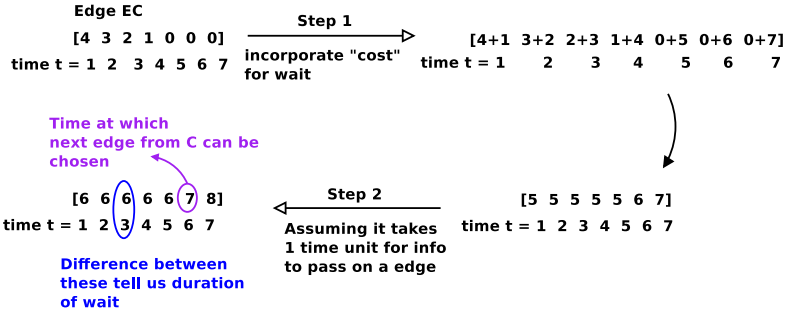

Fig. 11. Second step of earliest relay transformation

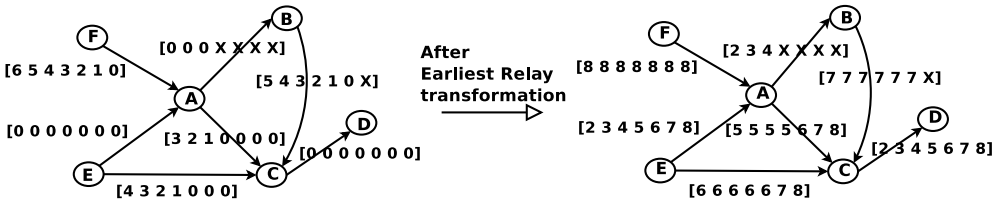

Fig. 12. Social event data after Earliest relay transformation

our previous example with edge (E,C), we would deem it to be non-existent at $t = 1$ as the duration of wait would be 4 time units. Note that this operationalization allows us to even have specific info-lifetime parameters for each individual or a message type. However, in this paper we restrict ourselves to single global value of info-lifetime parameter for ease of presentation. Now, the *Info-lifetime parameter* is used while exploring an edge for the shortest path. If the wait on an edge is more than the *Info-lifetime parameter*, it is not considered while computing the shortest path. Figure 12 shows the network from Figure 4 after the ERT transformation. We now define and describe our method of computing the *flow epoch points*.

4.2. Flow epoch based techniques

Flow epoch point: *A start time instant when the shortest path tree of a node may change.*

Consider again the problem of computing InLife betweenness centrality (Equation 1) on the social event data shown in Figure 12. A key component of this computation would be determining the shortest path between all pairs of nodes at all the times. As mentioned previously, these shortest paths could change with time. For instance, in our previous example with source node E, the shortest path between E and C (and thereby also for D) changed at $t = 5$. As soon as the shortest path to a node in the tree changes, we deem the shortest path tree to have changed. Thus, $t = 5$ becomes a *flow epoch point*. Now, in order to determine flow epoch points, we need to model the total cost of a path. This paper proposes a weight function to capture the total cost of a candidate path. This weight function, called a *path-function*, represents the earliest relay time from the end node of the path. A formal definition of the path-function is

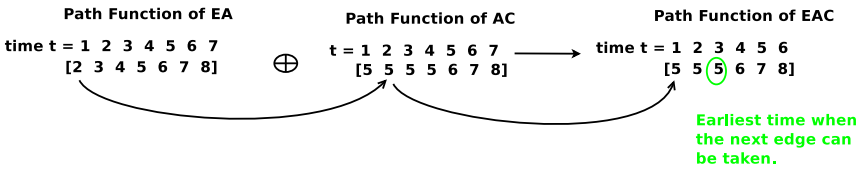

Fig. 13. Path function computation.

given below.

Path-function: *A path function is a time series that represents the earliest relay time from the end node of path as a function of time. A path function is determined by performing a composition of earliest relay time series of its component edges.*

For example, consider the path <E,A,C> in the social event data after the Earliest relay transformation in Figure 12. This path contains two edges, (E,A) and (A,C). The ERT time series of edge (E,A) is [2 3 4 5 6 7 8], while the ERT time series of edge (A,C) is [5 5 5 5 6 7 8]. Now, the path function of <E,A,C> for times [1,6] is computed as follows. If E sends an information unit at time $t = 1$ through edge (E,A), we would be able to use the next outgoing edge from A at time $t = 2$ because the value of the earliest relay time of edge (E,A) at time $t = 1$ is 2. Therefore, the next edge in path <E,A,C>, which is (A,C), can be used at time $t = 2$. Now, the earliest relay time of edge (A,C) time $t = 2$ is 5, making the next relay time from C to be 5. Now, C is the end node of our example path <E,A,C>. Therefore, we would stop the composition process and assign the value of 5 to the path function of <E,A,C> at time $t = 1$. The values of the path function for other start times is computed in similar fashion. This would give the path function of <E,A,C> as [5 5 5 6 7 8]. This means that if we send information from node E at times $t = 1, 2, 3, 4, 5, 6$ then the earliest next relay time from node C would be 5,5,5,6,7 and 8 respectively. This process is illustrated in Figure 13. Similarly, the path function of path <E,C> would be [6 6 6 6 6 7 8] (since it contains only one edge). A comparison of the two path functions would reveal that path <E,A,C> is shorter for times $t \leq 4$ and path <E,C> is shorter for times $t \geq 5$. Thus, in this case $t = 5$ would be a flow epoch point. In general, flow epoch points are determined by computing the earliest intersection point between the path functions. Here, the intersection between path functions of <E,A,C> and <E,C> is at $t = 5$.

5. FLOW-EPOCH-POINT BASED INLIFE BETWEENNESS CENTRALITY SOLVER

This section describes our Flow-epoch-point based algorithm (FIL-BETS) for computing the InLife betweenness centrality given by Equation 1. In order to use the efficient path counting technique proposed in [Brandes 2001], we generalize their notion of *pair-dependencies* to a temporal case. Thus, Equation 1 can be rewritten to interpret the centrality score of a vertex as a sum of *temporal pair-dependencies* (see Equation 2).

$$C_B(v)[t] \;=\; \sum_{s \neq v \neq d \in V} \delta_{sd}(v)[t] where, \quad \delta_{sd}(v)[t] \;=\; \frac{\sigma_{sd}(v)[t]}{\sigma_{sd}} \tag{2}$$

This is further abstracted in Equation 3 to create a notion of temporal dependency of a vertex $s \in V$ on $v \in V$. Now, in order to determine the centrality score we need to compute the value of δ, given by Equation 3, for each node $s \in V$ at all times $t \in \lambda$.

$$\delta_{s\cdot}(v)[t] \;=\; \sum_{\forall d \in V} \delta_{sd}(v)[t] \tag{3}$$

A key challenge in computing this would be the need to re-compute the shortest path tree (one for each time instant). We address this challenge by efficiently determining the *flow epoch points* when the shortest path tree of a node changes. Recall that these flow epoch points would partition the given time interval into a set of disjoint sub-intervals, over which the shortest path tree does not change. The *sub-interval temporally-ordered SPtree* denotes this shortest path tree.

sub-interval temporally-ordered SPtree: *is a shortest path tree, SPT_v, from a specific node, v, and its corresponding set of time instants ω_v. Here, any directed path from v to any of the nodes in SP_v is a shortest temporally-ordered path for all the time instants in ω_v. In other words, the shortest path tree SP_v is guaranteed not to change in ω_v.*

For example, the shortest path tree from the source node E containing paths, <E,A>, <E,A,B>, <E,A,C> and, <E,A,C,D> is a *sub-interval temporal ordered SPtree* with the corresponding $\omega_E = 3, 4$.

5.1. FIL-BETS Algorithm

The Flow-epoch-point based InLife BETweenness centrality Solver (FIL-BETS algorithm) consists of two key features. First, it efficiently computes the flow epoch points *on the fly* without enumerating all the possible candidate paths. We do this by using a search and prune strategy similar to Dijkstra's algorithm. This ensures bounded exploration of the space of possible candidate paths while ensuring correctness at the same time. Second, it employs a suitable adaptation of the path counting technique proposed in [Brandes 2001] to count the number paths passing through any particular node. We now provide a detailed description of the FIL-BETSalgorithm and illustrative execution trace on the dynamic social network data shown in Figure 12.

The algorithm starts by computing the shortest path tree of a node for the first time instant in λ. Due to the non-stationary behavior of dynamic social network data, the choice of the path to expand at each step made while computing the shortest path tree for one time instant may not be valid at later time instants. Therefore, the algorithm stores all the flow epoch points observed while computing the shortest path tree for one time in a data structure called a *path intersection table*. As discussed previously, the flow epoch point is determined by computing the time instants when the path functions of the candidate paths intersect. The earliest of these flow epoch points represent the first time instant when the current shortest path tree is no longer valid. Therefore, the algorithm restarts the computation from this time instant. Since, these path functions represent the earliest relay time for a information unit starting at the source, the intersection points would implicitly represent the time instants (at source) when the ranking of candidate paths change. This continues until shortest path trees are found for all the time instants in λ. The algorithm repeats the same procedure for all nodes in the network. In addition to storing the flow epoch points, the algorithm also maintains another data structure called *Predecessor table*. This table stores the predecessor(s) of every node in the shortest path tree from the current source node. Again, due to the non-stationary behavior of the social event, we store the predecessor information for each time instant.

The pseudocode of FIL-BETS is shown in Algorithm 1. The outermost for-loop (line 2–27 in Algorithm 1) ensures that the shortest path tree from all the nodes is determined. Within this for loop, the algorithm computes the shortest path tree of a single

node v for all the time instants in λ. This is done in the while-loop on line 5. Before this we initialize the two variables, $cur\text{-}start\text{-}time$ and $upper\text{-}start\text{-}time$ to the first and last start time in λ respectively. The while-loop on line 5 executes until the value of $cur\text{-}start\text{-}time$ is less than $upper\text{-}start\text{-}time$. The inner most while-loop in line 7 determines a single ***sub-interval temporally-ordered SPtree*** for the source node v. Here, we first initialize a priority queue, PQ, with path functions corresponding to immediate neighbors of source node. The while loop in line 7 runs until PQ is not empty. In each iteration, the path function having the least cost for $cur\text{-}start\text{-}time$ (pf_{min}) is extracted and its intersection with the other path functions in the PQ is computed. The earliest of these intersection points, t_{min}, is stored in the ***path intersection table***. Note that in the absence of intersection points, the value of t_{min} is set to the last time in λ.

After the value of t_{min} is determined, the algorithm determines all the possible predecessors of the tail node, min_{tail}, of the path corresponding to pf_{min}. This is accomplished as follows: We first choose all the paths in the PQ whose tail node is same the as min_{tail}. Now, the second to last nodes of all these paths are potential predecessors of min_{tail}. Among these candidates the ones who have the same cost as that of pf_{min} are stored as predecessors of min_{tail} (at corresponding time instants). For example, consider a path <s,x,y> whose corresponding path function [1 2 4 6 6] was chosen as pf_{min}. Further, consider another path function [1 2 7 7 7], corresponding to path <s,z,y>, present in the priority queue. For convenience, assume that the cur-star-time was $t = 1$ (corresponding to the first value in the path function) and $t_{min} = 4$. Here, the node z would be included in predecessors of node y for times $t = 1, 2$. Whereas, node x would be included for times $t = 1, 2, 3, 4$. This process is accomplished in lines 11–17 of the Algorithm 1. After the predecessors are determined, other paths ending on pf_{tail} are deleted from PQ and pf_{min} is expanded to include each of its neighbors. While expanding, the algorithm considers only those neighbors of pf_{tail} where wait duration is below the threshold specified by the *Info-lifetime parameter*. It may happen that an edge being considered now may not exist at a later time (e.g., edge (A,B) in Figure 12 does not exist after time $t = 2$) or an edge discarded now (due to long wait duration) may reappear later. We address both these cases by considering the corresponding time instants (when the edge disappears or reappears) while determining the next start time in line 22. If necessary, $cur\text{-}start\text{-}time$ is set at these time instants if they happen to be earlier than $t_{min}s$ in the path intersection table. Also, in the first case the predecessors are recorded accordingly in lines 11-17.

The newly determined path functions on line 19 are inserted into PQ. This process continues until PQ is empty. Note that when pf_{min} is expanded, pf_{tail} is closed for $cur\text{-}start\text{-}time$. This means that there cannot be any other path leading to pf_{tail} from the source with a shorter length.

After exiting from the inner while loop on line 7, the algorithm performs three functions. First, $cur\text{-}start\text{-}time$ (next start time) is set to the minimum of $t_{min}s$ stored in the path intersection table (line 22 in Algorithm 1). In a worst case scenario, this value could just be the next time instant in λ. In such a case, the sub-interval temporally-ordered SPtree determined by the inner most while loop would be valid for only one time in λ. Second, the predecessor table is reinitialized for times greater than the $cur\text{-}start\text{-}time$. Third, the dependency of each of the nodes (given by Equation 3) is computed for times between the previous start time $cur\text{-}start\text{-}time$. This is done using the recursive formulation proposed in [Brandes 2001].

5.2. Execution trace

We now provide an execution trace of the FIL-BETS on the dynamic social network (after earliest relay transformation) shown in Figure 12. Here, the set $\lambda = 3, 4, 5, 6$ and the value of the Info-lifetime parameter is 2. For brevity, we only show the computation

Algorithm 1 FIL-BETS Algorithm

1: Determine the earliest relay time series of each edge
2: **for all** nodes $s \in V$ **do**
3: $cur\text{-}start\text{-}time \leftarrow$ first start time in λ
4: $upper\text{-}start\text{-}time \leftarrow$ last start time in λ
5: **while** $cur\text{-}start\text{-}time \leq upper - start - time$ **do**
6: Initialize a PQ with the path functions corresponding to neighbors of s
7: **while** PQ is not empty **do**
8: Choose the path function, pf_{min}, with the minimum weight to expand
9: $t_{min} \leftarrow$ earliest intersection point of pf_{min} with path functions in PQ
10: Save t_{min} in the path intersection table
11: **if** multiple paths in PQ ending on min_{tail} **then**
12: **for all** time instants t in interval $cur - start - time$ through t_{min} **do**
13: Set the second last node of all the paths (including pf_{min}) which have the same cost pf_{min} for time t as predecessor min_{tail}
14: **end for**
15: **else**
16: Set the second last node in pf_{min} as the predecessor of min_{tail} for all times $cur\text{-}start\text{-}time$ through t_{min}
17: **end if**
18: Delete all the paths ending on pf_{tail} from PQ
19: Determine the path functions resulting from expansion of the chosen path
20: Push the newly determined path functions into PQ
21: **end while**
22: $cur\text{-}start\text{-}time \leftarrow \min(t_{min}$ in path intersection table)
23: Clear path intersection table
24: Re-initialize the predecessor table for times greater than $cur\text{-}start\text{-}time$
25: Compute the dependency of s on each vertex for times between previous start time and $cur\text{-}start\text{-}time$
26: **end while**
27: **end for**

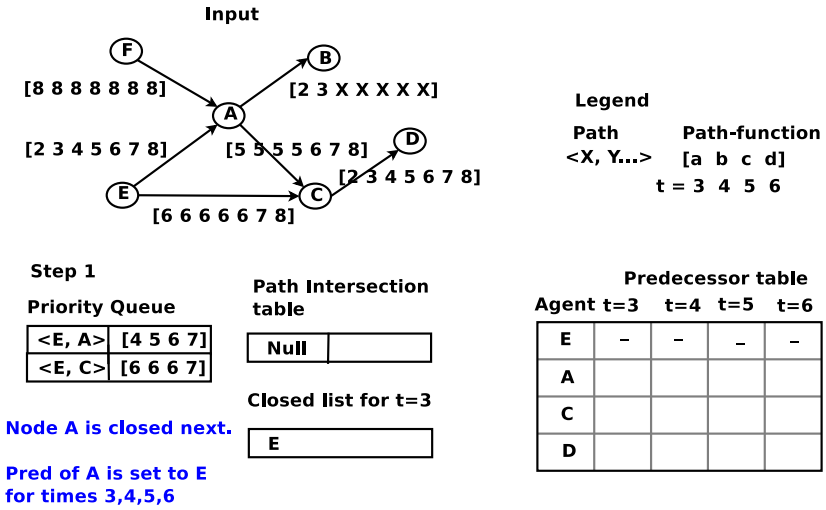

Fig. 14. FIL-BETS Execution trace Step 1

of a single sub-interval temporally-ordered SPtree with source E. Recall that FIL-BETS divides the set λ into a set of disjoint sub-intervals, where each sub-interval is associated with a shortest path tree called a sub-interval temporally-ordered SPtree.

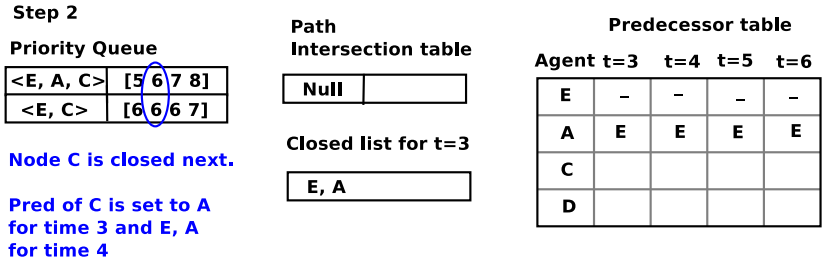

Fig. 15. FIL-BETS Execution trace Step 2

In the first iteration of the while loop in line 5 the value of $cur\text{-}start\text{-}time$ would be 3 (first time instant in λ). The inner while on line 7 begins with a priority queue, PQ, initialized with the path functions corresponding to immediate neighbors of E (see Figure 14). Here, path functions corresponding to both A and C (neighbors of E) are defined for the entire range of λ. Now, the path function having the lowest cost at $t = 3$ would be chosen as pf_{min} in line 8. In this case the path <E,A> has the lowest cost for $t = 3$. Since the path function of <E,A> has no intersection points with other paths in PQ, the value of t_{min} is set to the last time in λ. Now, since there are no paths ending with node A in PQ, E would be set as the predecessor of A for times $3, 4, 5, 6$. Lines 20-21 would expand the path <E,A> to insert the path function corresponding to <E,A,C> (see Figure 15).

Second iteration of the while loop on line 7, would select the path <E,A,C> as pf_{min} and node C would be closed for time $t = 3$. Also, there is an intersection of the path function of <E,A,C> with <E,C> at $t = 5$ (see Figure 15). This intersection point would be stored in the path intersection table. Now, PQ has two paths <E,C> and <E,A,C> (see Figure 15) ending at node C. Therefore, the path functions of these two paths are compared to decide the predecessors of C. The algorithm assigns A as the predecessor of C for $t = 3$, while, both E and A are included as predecessors of C at $t = 4$ (see Figure 16). The algorithm then expands the path <E,A,C> and the path function corresponding to <E,A,C,D> is inserted into PQ. Other paths ending on node C are deleted from PQ.

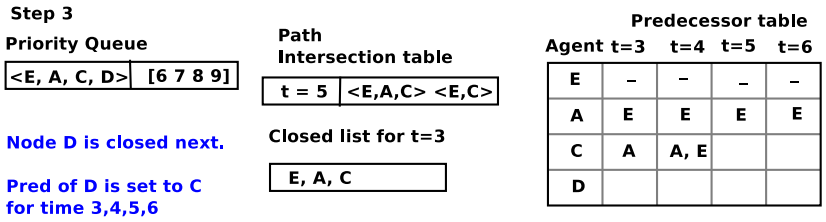

Fig. 16. FIL-BETS Execution trace Step 3

In the last iteration of the while loop, the algorithm closes the node D for time $t = 3$ and its predecessor is assigned appropriately. Node D does not have any neighbors. Therefore, the priority queue would be empty. This is the termination condition of the while-loop on line 7. After the the termination of this loop, the next start time instant is determined by selecting the earliest of the $t_{min}s$ stored in the path intersection table. This happens to be $t = 5$ for this example (see Figure 17). Thus effectively, the

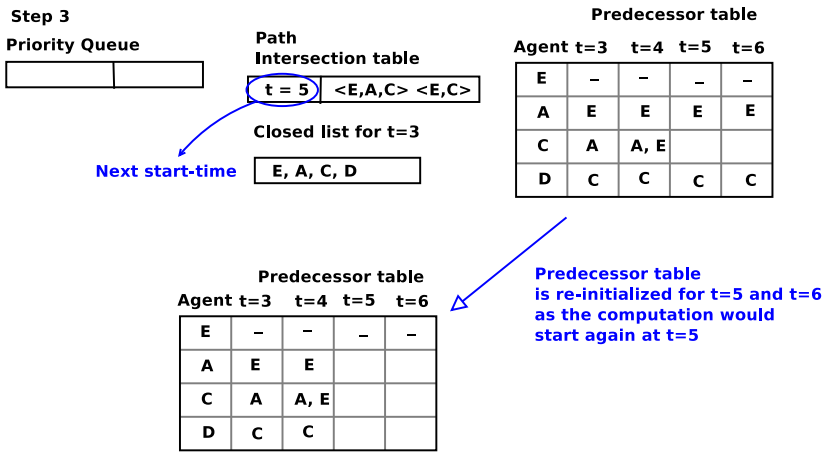

Fig. 17. FIL-BETS Execution trace Step 4

algorithm has now determined a sub-interval temporally-ordered SPtree with source E for times $t = 3, 4$. Note that we did not compute the shortest path tree for $t = 4$ separately.

The above process of computing the shortest paths will now repeat for $t = 5$, and the next start time would be determined accordingly. Before proceeding to the next iteration of the while-loop on line 5, the algorithm re-initializes the predecessor table for times greater than $t = 5$ (see Figure 17). After that partial dependencies (Equation 3) of all the nodes are computed in a recursive fashion as proposed in [Brandes 2001].

6. ANALYTICAL EVALUATION

The FIL-BETS algorithm divides a given interval (over which shortest path trees have to be determined) into a set of disjoint sub-intervals. Within these intervals, the tree does not change. The correctness of the FIL-BETS algorithm requires that the set of predecessors determined for any particular node (lines 11-17 of Algorithm 1) be comprehensive. On the other hand, the completeness of the algorithm guarantees that none of the flow epoch points are missed. Lemma 6.3 and Corollary 6.4 are used to show that the set of predecessors determined by FIL-BETS is comprehensive, i.e., the algorithm records all the potential predecessors of a node while building an sub-interval temporally-ordered SPtree. On the other hand, Lemma 6.1 shows that the FIL-BETS algorithm does not miss any flow epoch point. These lemmas are used to prove the correctness and completeness of the FIL-BETS algorithm.

LEMMA 6.1. *The FIL-BETS algorithm recomputes the shortest path tree (for any particular node $s \in V$) for all those start times $t_i \in \lambda$, where the tree for previous start time t_{i-1} can be different from that of t_i.*

PROOF. The shortest path tree of a node $s \in V$ is considered to have changed if the shortest path to any node t (from s) in the tree is different across times t_i and t_{i-1}. For the sake of brevity, we only give the proof for a single node t in the tree. This can be easily generalized for the whole tree.

Consider the network shown in Figure 18(a), where a shortest path tree of the node s is to be determined for each time instant in $\lambda = [1, 2 \ldots T]$. In this proof we focus on the shortest path to node d instead of the whole tree. First, the source node is expanded

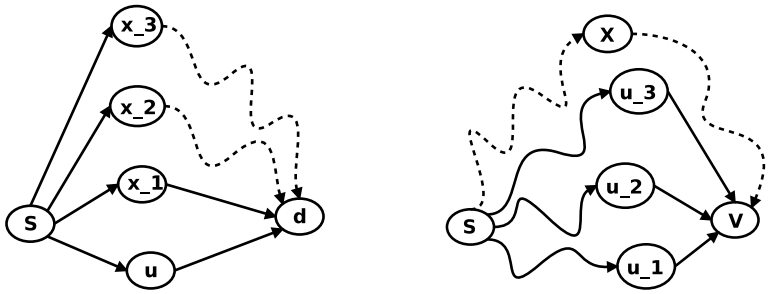

Fig. 18. Sample network for Lemma 6.1 and Lemma 6.3

for the start time instant $t = 1$. As a result, path functions for all the neighbors <s,u>, <s,x_1>, <s,x_2>, ... <s,x_i> are added to the priority queue. With loss of generality assume that path <s,u> is chosen in the next iteration of the innermost loop. Also assume that the earliest intersection point between <s,u> and the path functions in the priority queue is at $t = \alpha$ (between <s,u> and <s,x_1>). After expanding <s,u> the queue would contain paths <s,u,d>, <s,x_1>, ..., <s,x_i>. Here we can have two cases. First, path <s,u,d> has lower cost for start time $t = 1$. Second, path <s,x_1> has lower cost for start time $t = 1$.

Consider the first case, again without loss of generality assume that the earliest intersection point between the path functions <s,u,d> and <s,x_1> is at $t = \beta$. Note that both $t = \alpha$ and $t = \beta$ denote the start times at the source node. Now, $\beta \leq \alpha$ (note that β cannot be greater than α as all the edges have positive weights). In such a case the shortest path is recomputed for starting time $t = \beta$ and the path <s,u,d> is closed for all start times $1 \leq t < \beta$. Assume for the sake of argument, that there is a shortest path $P_x =< s, x_2, \ldots, d >$ from s to d that is different from path <s,u,d> and is optimal for start time $t_x \in [1, \beta)$. This means that path $< s, x_2 >$ should have had least cost for time $t = t_x$. In other words, the earliest intersection point between functions in the priority queue and $< s, u, d >$ should have been at $t = t_x$ rather than at $t = \beta$ (a contradiction). Moreover, as all the travel times positive, if sub path <s,x_2> was not shorter than <s,u,d> for start times $t \in [1, \beta)$. Any positive weight addition to the path function (through other edges) cannot make P_x shorter than <s,u,d> for $t \in [1, \beta)$

Now we consider the second case when <s,x_1> had lower cost for start time $t = 1$. Now, path <s,x_1> would be expanded and path function <s,x_1,d> would be added to priority queue. A similar argument as above can be given for this case as well. □

COROLLARY 6.2. *The FIL-BETS algorithm correctly computes the shortest path tree (for any particular node $s \in V$) for any particular time instant $t \in \lambda$.*

LEMMA 6.3. *The set of predecessors determined for a node v for a time $t \in \lambda$ is complete.*

PROOF. Let $P(v) = P_{u_1}(v), P_{u_2}(v), P_{u_2}(v), \ldots, P_{u_k}(v)$ denote the set of paths currently in the priority queue whose tail node is v. Here, P_{u_i}, where $i \in [1, k]$, denotes the path whose second-to-last node is u_i (and last node being v) and $cost(P_{u_i}(v))$ denotes the cost of path $P_{u_i}(v)$ for time $t = \alpha$. Without loss of generality assume that the node v is being closed for time $t = \alpha$ (see Figure 18(b)) through the path P_{u_1}, i.e., P_{u_1} is found to be a shortest path for node v. Now, the FIL-BETS algorithm assigns a node

u_x as a predecessor of node v based on the following cases;

Case 1: u_x is the second-to-last node in the shortest path found, i.e., $u_x = u_1$.
Case 2: $u_x \in U$, where $U = \{u_i | cost(P_{u_i}(v)) = cost(P_{u_1}(v)) \& i \neq 1\}$.

In order to prove the lemma, we show that the above two cases are correct and complete. We first show the correctness followed by completeness. Using Corollary 6.2 we know that when a node v is closed by the FIL-BETS algorithm for any particular time t, it has correctly determined the shortest path for v. Therefore, the node preceding v in this path is clearly a predecessor of time t. This proves the correctness of Case 1. Correctness of Case 2 can be also guaranteed in similar fashion as we include the second-to-last nodes of only those paths whose cost is same as that of the shortest path found by FIL-BETS.

In order to show the completeness, we need to prove that there cannot be any other path Q, not currently in the priority queue, with $cost(Q) = cost(P_{u_1})$. For sake of argument assume that there exists a path $Q =< s, \ldots, x, v >$, not currently in PQ, and x is valid predecessor of v for time $t = \alpha$. By definition the distance of x from s should be less than that of v (as we have only positive weights). This means that node x should be closed before v. Now, whenever a node is closed by the algorithm, path functions corresponding to its neighbors are added to the priority queue. This contradicts our original assumption that path Q was not present in the priority queue. □

COROLLARY 6.4. *The set of predecessors determined for a node v for all times $t \in \lambda$ is complete.*

PROOF. Using Lemma 6.1, we know that FIL-BETS does not miss any flow epoch point. Now, during the time interval between two consecutive flow epoch points, Lemma 6.3 can be used to conclude that the predecessors are correctly determined for all time time instants within the interval. Using these two arguments we can conclude that FIL-BETS correctly determines the predecessors of a node v for all times $t \in \lambda$. □

THEOREM 6.5. *FIL-BETS algorithm is correct and complete.*

PROOF. For any particular node v, there can be several sub-interval temporally-ordered SPtrees, $SPT_v = \{SPT_v^1, SPT_v^2, \ldots, SPT_v^k\}$, over the λ, where each SPT_v^i is associated with a set of time instants ω_v^i and $\bigcup_{\forall i \in |SPT_v|} \omega_v^i = \lambda$. The correctness of the algorithm is shown by Lemma 6.3 and Corollary 6.4. The completeness proof is presented in three parts. First, using Lemma 6.1 we can conclude that the FIL-BETS algorithm does not miss any flow epoch point. Secondly, the loop on line 5 of the algorithm ensures that all time instants in λ are considered. Finally, the outer most loop on line 2 ensures that all the nodes $v \in V$ are processed. This proves the completeness of the algorithm.

□

Discussion: The FIL-BETS algorithm would perform less computation than a naive approach which determines the shortest path trees of a node for each start time in the user specified time interval. However, this happens only when the ratio of the number of flow epoch points to that of start time instants is low. This ratio can be denoted as the change probability shown by Equation 4.

$$change\_probability = \frac{\#flow\ epoch\ points}{\#start\ time\ instants in \lambda} \tag{4}$$

When the change probability is nearly 1, there would be a different shortest path tree (of a node) for each start time in an interval. In this worst case scenario, the FIL-BETS approach would also have to recompute the shortest path tree for each start time. A theoretical bound of $n^{\Theta(\log n)}$ (where n is the number of nodes in the graph) was given in [Foschini et al. 2011] on the number of flow epoch points for the case of piecewise linear cost functions. However, in our case the number of flow epoch points is bounded by length of start time interval ($|\lambda|$) due to discrete nature of the input.

7. EXPERIMENTAL EVALUATION

Experiments were conducted to evaluate the performance of the FIL-BETS as different parameters were varied. The experiments were carried out on a real dataset containing email communications spread over 83 days in early 2004 from a large European university. This dataset recorded about 161227 email communications among approximately 2000 individuals. Events in this dataset (email communications) were recored in the following format: $<A,B,t>$, where A and B are two individuals in the University and t denotes the timestamp (in minutes) of the interaction. In terms of explicit info-path models, this dataset would be represented as a graph with about 235 million nodes and edges. Whereas, our time-aggregated graph based model had only 2000 nodes and 24000 edges, where each edge was associated with a time series of length approximately 117,500. Experiments were carried out on a Linux workstation with Intel Xeon Processor and 8GB RAM. The performance of FIL-BETS was compared against a modified version of single a start time based shortest path algorithm proposed in [George et al. 2007] called *SP-TAG*. We now briefly describe this approach.

Modified SP-TAG (MSP-TAG) Algorithm: The MSP-TAG algorithm is a label setting algorithm to compute shortest path trees for a single time instant. This algorithm uses a search and prune strategy, similar to Dijkstra's, to explore the space of candidate paths. Given a source s, start time instant t, and an *info-lifetime* parameter, MSP-TAG would return a tree rooted at s containing shortest paths to all the other nodes. These paths have two characteristics: (a) edges in the paths are temporally ordered; (b) max wait duration at a node is less than the given info-lifetime parameter. Along with the shortest paths, it also also returns the list of predecessors for all the nodes. This predecessor list is post processed using [Brandes 2001] to return InLife betweenness values. The primary reason for choosing this method was to enable a direct comparison of our *flow epoch* based approach against a naive version which computes centrality values for each time instant in λ.

Experimental setup: The experimental setup is shown in Figure 19. The first step of the experimental evaluation involved creating a time aggregated graph out of the dynamic social event data. Then, a set of different queries (each with a different set of parameters) was run on the FIL-BETS and the MSP-TAG algorithms. The following parameters were varied in the experiments: (a) Info-Lifetime parameter, (b) $|\lambda|$: Length of the time interval over which the centrality metric was computed, (c) Network size. In each of the experiments the total execution time of both the algorithms was recorded.

Effect of Info-Lifeitme parameter: We tested the effect of increasing the *Info-lifetime* parameter on the candidate algorithms. As shown in Figures 20 and 21, runtime for both the algorithms increased with a longer information lifetime. However, runtime of FIL-BETS increased at a slower rate than that of MSP-TAG.

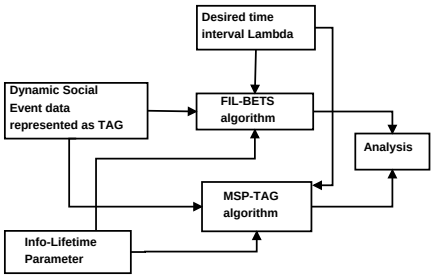

Fig. 19. Experimental setup.

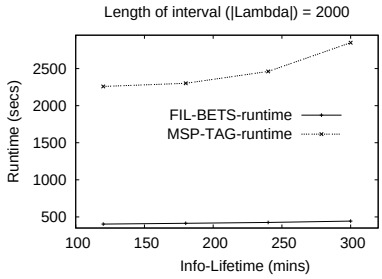

Fig. 20. Effect Info-Lifetime parameter for $|\lambda| = 2000$.

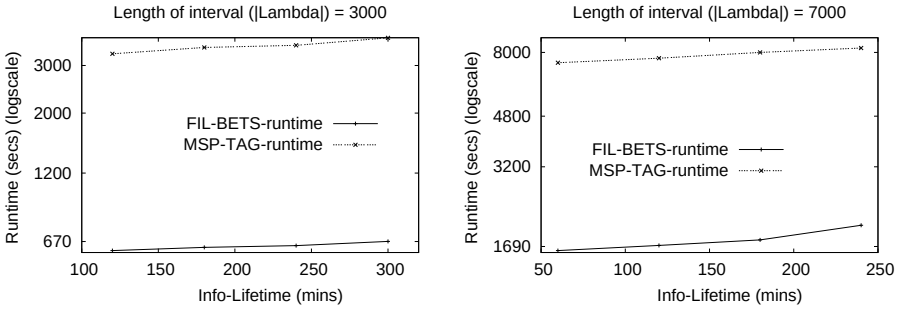

Fig. 21. Effect Info-Lifetime parameter for $|\lambda| = \{3000, 7000\}$.

Moreover, for any particular value of *Info-Lifetime* parameter, the FIL-BETS algorithm was faster than MSP-TAG algorithm by several orders of magnitude. This is primarily because FIL-BETS efficiently avoids recomputation of shortest paths for certain time instants without compromising the correctness of the solution, unlike MSP-TAG, which must recompute shortest paths for all the time instants in the given time interval λ.

Effect of Length of time interval ($|\lambda|$): This experiment evaluated the effect of length of time interval (λ) on the candidate algorithms. The results, shown in Figure 22, indicate that execution time of both the algorithms increased linearly with increase in $|\lambda|$. However, for any particular value of $|\lambda|$ and Info-lifetime value, FIL-BETS was faster than MSP-TAG by an order of magnitude.

Effect of Network size: We tested the algorithms on three different size networks. To create the networks, we removed some nodes and their edges from our original dataset (referred to as Dataset3) to created two smaller datasets called Dataset1 and Dataset2. Dataset1 had 1095 nodes and 18000 edges, whereas Dataset2 had 1300 nodes and 21000 edges. In both the smaller datasets the length of the time series was approximately 100,000. The algorithms were tested on all the three datasets. We also varied info-lifetime parameter and length of lambda. Figure 23 shows the effect of the Network size on the performance of the algorithms. As can be seen the execution of time of both the algorithms increased with the increase in network size, but FIL-BETS still outperforms MSP-TAG.

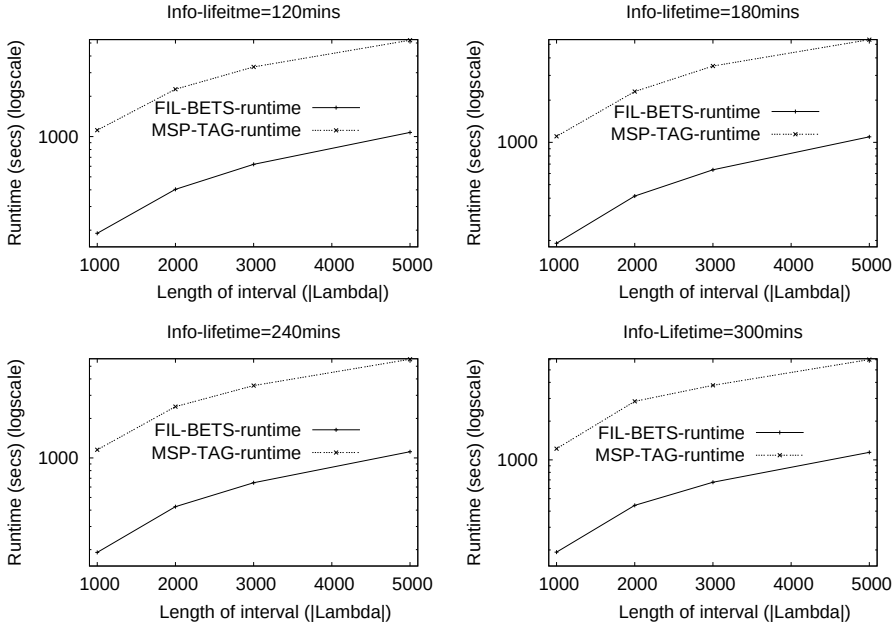

Fig. 22. Effect of length of time interval ($|\lambda|$).

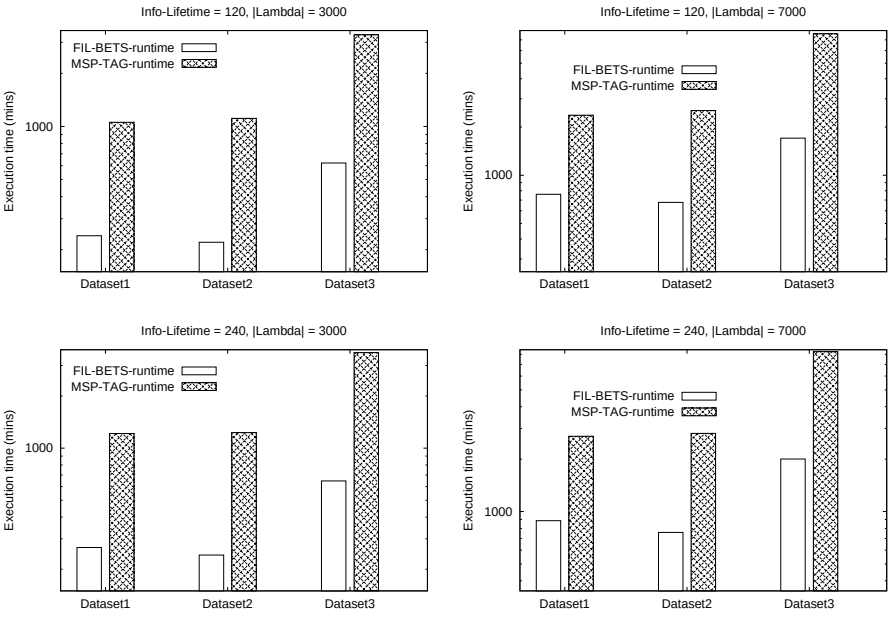

Fig. 23. Effect of Network size.

8. CASE STUDY

Having described the computational advantages of our approach over previous efforts, we now turn to a case study on real-world social interactions. We will first describe the data used for the case study and the cleaning methodology. We then focus on how the results of InLife betweennss centrality differ from traditional betweenness centrality computed on aggregated panel data. We also show how the InLife betweenness centrality reflects the an individuals circadian rhythms (or diurnal cycle).

8.1. Data and Cleaning

The data we use is a set of email communications spread over eighty three days in early 2004 from a large European university. The dataset has been used in several previous studies [Eckmann et al. 2004; Barabási 2005; Malmgren et al. 2009; Malmgren et al. 2008], with a focus on both the prevalence of dynamic triads [Eckmann et al. 2004] and on characterizing the inter event dynamics of users (i.e. the time between two consecutive email sends) [Barabási 2005; Malmgren et al. 2009; Malmgren et al. 2008]. Though we will utilize findings from these works extensively, there are two preliminary observations from them specific to data collection which we find necessary to note here. First, as stated by [Malmgren et al. 2009], based on communication with [Eckmann et al. 2004], the data was collected "before home Internet access was common, thus it is likely, for example, that activity on weekends is systematically lower than for an equivalent contemporary population". This claim suggests that properties of this dataset regarding the importance of the work day in information flow may be exaggerated as compared to more recent works. Second, as noted in [Malmgren et al. 2009; Barabási 2005; Eckmann et al. 2004], the dataset does contain several nodes which appear to be mail servers as well as several people who do not communicate often enough. We choose to remove such nodes from the study.

8.2. How does InLife betweenness centrality compare with the traditional betweenness centrality calculated on aggregated panels?

[Eckmann et al. 2004] notes, when studying the data utilized in this case study, that "temporal dynamics create coherent space-time structures that...will in general be very different from the fixed ones that appear in the static network". However, [Eckmann et al. 2004] is in this quote referring to the network created by aggregating all the events into a single panel, i.e., the network comprised of all eighty three days of interaction data. In contrast, our interest here is in comparing the results of our algorithm to panels obtained by aggregating over shorter time intervals.

One disadvantage to our approach is that, while it is straightforward to determine which agent is the most central at any given time instant (a minute, in this case study), it is not immediately obvious how one should determine which agent is the most central over an entire time interval. This is because at each time instant, all agents may be on some number of shortest paths, and thus have some non-zero betweenness centrality score. Therefore, if we wish to determine which agent was the most central on a given day, we must somehow aggregate over all InLife betweenness values, for all agents, for each minute in that day. Though our approach was not geared towards such an cumulative notion of centrality, a quantitative comparison between our method and the aggregated panel is necessary and thus some calculation of the ranks of different agents for betweenness over a given aggregation must be computed.

The manner in which we choose to do this is used for it simplicity and explainability, though others no doubt exist which may be more powerful along one or both of these dimensions. In order to determine which agent is most central over a given period of interval, we first determine the most central agent (i.e. the one with the highest

InLife centrality score) at each time instant (minute), and then aggregate over the desired time period (e.g. a day) to produce a count, for each agent, of the number of time instants he/she was most central for. The rank of a node (e.g. most central, second most central, etc.) for the given time interval is then determined by his or her rank in this list of counts.

Before we can compare to some betweenness metric on aggregated panels of the network, we must also determine what representation of the network each panel will have. [Johnson et al. 2012] considers different meanings of an edge in an email network, including removing mass emails to get a more social view of the network. Here, because we consider betweenness as an information flow metric, we choose to keep mass emails in the data. On a different note, [Barabási 2005] sees that most users receive more emails than they send, suggesting further that, of course, agents thus receive far more emails than they respond to. This finding suggests email communication may be particularly susceptible to differences in directed versus undirected measures of betweenness, and hence we use a directed approach in order to compare best with our algorithm.

Having operationalized the notion of an agent being "most central" over any given time interval across both methods, we now may compare the results of our metric with the traditional betweenness centrality on aggregated panel. We compare across five different levels of aggregation: one hour, two hours, ten hours, one day and three days. In full, the process for comparing the metric at a single level of aggregation (e.g. a day) can be described in three steps. We first aggregate the interaction data into panels and then compute the traditional *weighted, directed* betweenness centrality [Freeman 1979; Anthonisse 1971] for agents in each using the igraph package in R [Csardi and Nepusz 2006]. We then use the method described above to compute the centrality of each agent within each of the aggregated intervals for the InLife betweenness centrality metric. Finally, we utilize one of three comparison metrics, first introduced by [Borgatti et al. 2006] and later used by [Frantz et al. 2009], to compare agreement between the two approaches for each set of aggregated interactions.

The three comparison metrics we use are "Top 1", "Top 3" and "Top 10%" (the original names provided by [Borgatti et al. 2006]. "Top 1" refers to the percentage of panels over which the most central node is the same across both techniques. "Top 3" refers to the percentage of panels that the top node using the traditional approach on aggregated panel is one of the top three nodes utilizing our metric. "Top 10%" refers, similarly, to the percentage of snapshots in which the top node in the traditional approach on aggregated panel approach is one of those in the top ten percent using our metric. We choose these metrics over other similar metrics [Kim and Jeong 2007] because they are straightforward to understand while still providing depth for intuition. In collecting results, we only consider as input those panels where at least one node was central using both approaches and where there were no ties as to the most central node in the aggregated panel network. It is also important to note that at lower Info-lifetime values and aggregations, because often less than 30 nodes have some non-zero centrality value, our results suggest the counter-intuitive finding that in some cases, "Top 3" shows a higher percentage of agreement across the two algorithms than "Top 10%". Also, because larger aggregation levels naturally have fewer data points, the reader will notice that confidence intervals surrounding the point estimates for these metrics naturally increase with aggregation.

Figure 24 shows the results for each of the three comparison metrics, where points represent the point estimate and error bars give the 95% confidence intervals using Wilson's formula [Agresti and Coull 1998]. The values on the X-axis represent different Info-lifetime values our algorithm was run with, while the values in the grey bars above each graph represent the level of aggregation, in minutes, over which the inter-

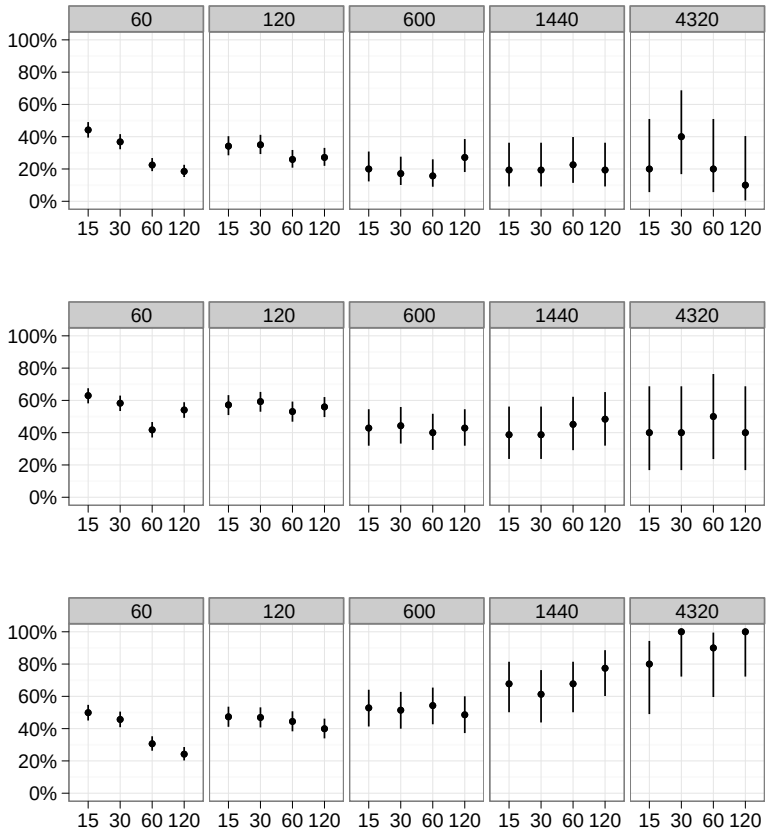

Fig. 24. Percentage of aggregation sets, across all Info-lifetimes (X-axis) and all levels of aggregation (separate plots, level of aggregation in minutes in grey bar) where the most central agent in the snapshot method is in the top one (top set of plots), top three (middle set) or top 10% (bottom set) of most central agents using our approach (best viewed in color)

actions were split. In considering the "Top 1" and "Top 3" metrics (the top two plots), we see that although there is reasonably high variance across individual combinations of Info-lifetime and aggregation level, there are several general patterns. First, agreement between the two methods is slightly higher at lower levels of aggregation. This finding is not particularly surprising; at lower levels of aggregation, there are fewer emails and less time to be aggregated over; resulting in fewer invalid temporal paths and fewer unrealistic durations for which information is held at a given node. However, at these levels, we do see that higher Info-lifetimes have much lower levels of agreement. This, also, has a fairly obvious explanation in that these Info-lifetime values consider data far outside the boundaries of the aggregation, and thus will likely present different pictures of the network. This explanation is further supported by the fact that this difference appears to slow at higher levels of aggregation.

Figure 24 thus suggests following conclusions. First, the agreement between the two metrics on the most central nodes (Top 1 comparison metric) is not more than 60%. Second, there is very high aggrement between the two metric in case of Top 10% over long periods of aggregatation. This shows that our metric does measure something which does not entirely agree with what is being measured in the aggregation based method. In other words, these results do show an inclination towards the fact aggregation based techniques may consider paths which are not feasible for information flow.

8.3. Does an individual's circadian rhythms play a role in his/her InLife betweenness centrality?

Even in restricting scope to papers written on the same dataset we study here, much has been done to understand the heavy-tailed distribution which governs the time between when a person sends an email and when they send again. This distribution, which has been fit in this dataset to the lognormal [Malmgren et al. 2008] and the pareto [Barabási 2005], has been of interest to complex system researchers for some time, and has been generally referred to as "interevent dynamics"; a term we will use again. While we have little interest in further pressing the question of the statistical formulation of interevent times, the fact that people tend to send a series of emails in rapid "bursts" is fundamental to our understanding of the patterns of centrality which our metric gives. As [Malmgren et al. 2008; Malmgren et al. 2009] note, one explanation for the burstiness observed in interevent dynamics is that email behavior (particularly, as we have noted, in this dataset) is affected by circadian rhythms. Thus, if our metric indeed captures the burstiness inherent to the data, we should observe that centrality scores are themselves affected by circadian rhythms.

Figure 25 plots a point for each agent which, at some point, highest in centrality during the month of data we observe. The graphs of different Info-lifetime values are separated by the grey bars above them, which signify the duration, in minutes, of the Info-lifetime parameter for those data points. On the X-axis, each day in the data is represented - a tick mark on the axis represents 05:00 on the stated day of the week - the scale of the data, however, is in minutes. Each increment of the Y-axis is a different agent - though difficult to discern which agent is most central at any given time, we do not focus on this in particular here. The color of a point, as represented by the scale on the right of the plot, represents the duration across which an agent was most central. Each point is placed at the beginning of the time the agent was central for.

Figure 26 is the same as Figure 25, but simply "zoomed in" to the first forty-eight hours of the dataset. These plots provide us with two useful points. First, our centrality metric appears to be affected by the same circadian rhythms as those which [Malmgren et al. 2008] postulate affect interevent dynamics. In particular, we note that for all Info-lifetimes below one hour, points are quite dense during the working hours, less prevalent during off-peak hours, and even less so on the weekend. This distribution is not a finding determined a priori by the setup of the algorithm but rather draws from the fact that emails which are both near enough in time to be considered a chain and which connect paths of longer than length one are simply quite rare in these off-peak hours.

Second, a higher Info-Lifetime parameter enables a agent to stay central for longer time periods. This is because higher Info-lifetime parameter allows information to "live" at a node for longer time duration. Consequently, emails which are separated by long time periods are also considered as part of the same information flow.

Regardless, in this section we have noted a possible correlation between circadian rhythms, inter event dynamics and InLife centrality in dynamic social networks. Such a claim suggests a hypothesis for future work, namely that those who are central in

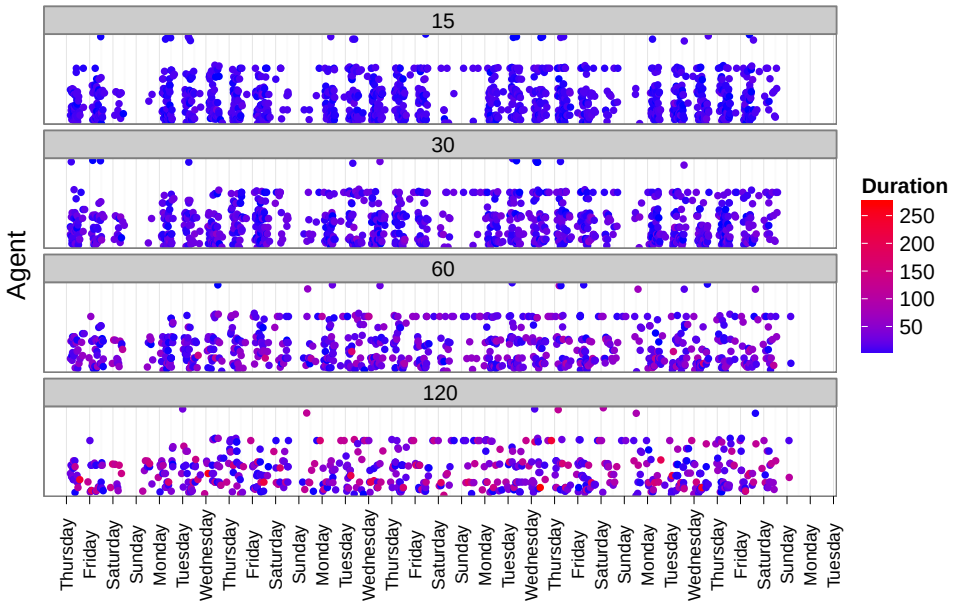

Fig. 25. Plot of the most central agents for the duration of the dataset (best viewed in color). The X-axis is time (each axis tick is 05:00 on the given day of the week), the Y-axis has a unique value for each agent which was ever central. Color of the point represents duration (in minutes), as shown in the legend to the right of the plot.

a dynamic approach such as ours not only are central in the information cascades but also have their circadian rhythms most in sync with others.

9. CONCLUSION

The problem of determining dynamic information liaisons (DINFL) is a important is several societal applications such viral marketting, word of mouth marketing and finding key influencers. However, finding DINFL poses both semantic and computational challenges. Semantic challenges arise due to inherent properties of the information pathways such as, bounded lifetime of information and Information flow frame of reference. Whereas, computational challenges arise due to the non-stationarity of dynamic social networks, leading to variability of number of shortest paths passing through an individual. Related work in the area of DINFL is limited due to their inability to address the semantic and computational challenges. The proposed InLife betweeness centrality metric addressed the semantic challenges DINFL by considering only those paths through which the same piece of information might feasibly move. We achieved in two ways. First, we considered only candidate paths in which the social events (e.g. emails interactions) are temporally ordered. Second, incorporated the notion of Information lifetime at a individual. The computational challenge was addressed through a novel computational structure called *flow epoch based approach*. The proposed computational technique enabled us to compute InLife betweenness centrality on large dynamic social networks. Theoretical and experimental analysis showed that this approach was several times faster than alternatives.

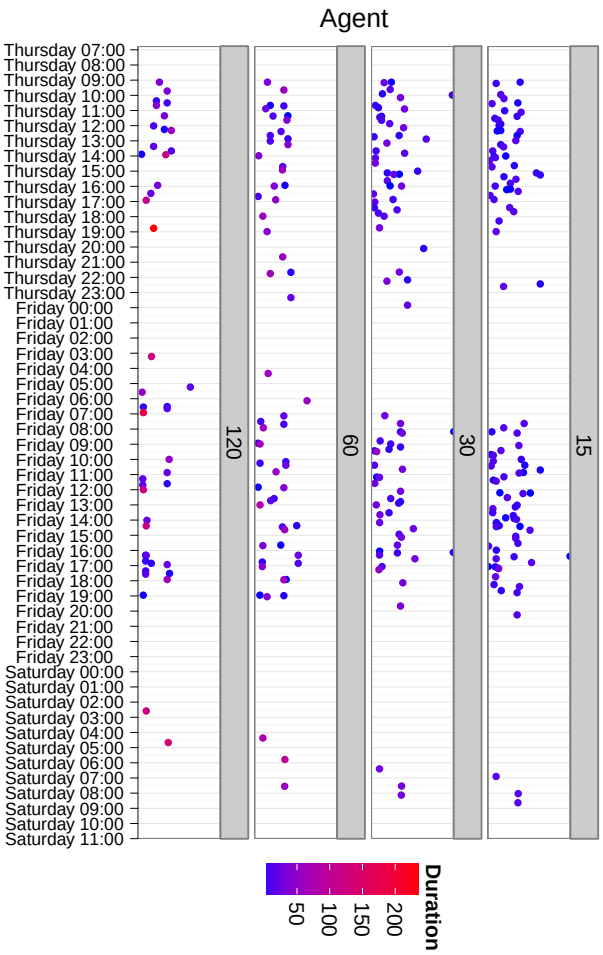

Fig. 26. The same as Figure 25, except zoomed in on the first 48 hours of the data (best viewed in color).

In the future we plan to extend our approach to include a probabilistic version of information lifetime (instead of a discrete value). We also plan to compare our results with other centrality metrics such as closeness centrality and α-centrality. Future experiments will consider other media beyond email data, possibly containing the contextual information (e.g. twitter feeds, interactions in online social networks etc).

Acknowledgment We are particularly grateful to the members of the Spatial Database Research Group at the University of Minnesota and Computational Analysis of Social and Organizational Systems at Carnegie Mellon University for their helpful comments and valuable suggestions. We would also like to extend our thanks to Kim Koffolt for improving the readability of this paper. This work was supported by NSF grants (grant number NSF III-CXT IIS-0713214, NSF No.1029711, NSF CRI:IAD

CNS-0708604) and USDOD grant (grant number HM1582-08-1-0017, HM1582-07-1-2035 and W9132V-09-C-0009). The content does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred.

REFERENCES

AGRESTI, A. AND COULL, B. A. 1998. Approximate is better than "Exact" for interval estimation of binomial proportions. *The American Statistician 52,* 2.

ANTHONISSE, J. M. 1971. The rush in a directed graph. CWI Technical Report Stichting Mathematisch Centrum. Mathematische Besliskunde-BN 9/71, Stichting Mathematisch Centrum.

BARABÁSI, A.-L. 2005. The origin of bursts and heavy tails in human dynamics. *Nature 435,* 7039, 207–211.

BARABSI, A. L., JEONG, H., NDA, Z., RAVASZ, E., SCHUBERT, A., AND VICSEK, T. 2002. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications 311,* 3-4, 590 – 614.

BORGATTI, S. P., CARLEY, K. M., AND KRACKHARDT, D. 2006. On the robustness of centrality measures under conditions of imperfect data. *Social Networks 28,* 2, 124–136.

BÖRNER, K., DALL'ASTA, L., KE, W., AND VESPIGNANI, A. 2005. Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams: Research articles. *Complex. 10*, 57–67.

BÖRNER, K., MARU, J. T., AND GOLDSTONE, R. L. 2004. The simultaneous evolution of author and paper networks. *Proceedings of National Academy of Sciences of the United States of America 101,* Suppl 1, 5266–5273.

BRANDES, U. 2001. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology 25,* 2, 163–177.

COLEMAN, J. S. 1988. Social capital in the creation of human capital. *American Journal of Sociology 94*, pp. S95–S120.

CSARDI, G. AND NEPUSZ, T. 2006. The igraph software package for complex network research. *InterJournal Complex Systems*, 1695.

DIESNER, J., CARLEY, K., AND TAMBAYONG, L. 2012. Extracting socio-cultural networks of the sudan from open-source, large-scale text data. *Computational & Mathematical Organization Theory 18*, 328–339.

ECKMANN, J.-P., MOSES, E., AND SERGI, D. 2004. Entropy of dialogues creates coherent structures in e-mail traffic. *Proceedings of the National Academy of Sciences of the United States of America 101,* 40, 14333–14337.

FOSCHINI, L., HERSHBERGER, J., AND SURI, S. 2011. On the complexity of time-dependent shortest paths. In *SODA*. 327–341.

FRANTZ, T. L., CATALDO, M., AND CARLEY, K. M. 2009. Robustness of centrality measures under uncertainty: Examining the role of network topology. *Comput. Math. Organ. Theory 15,* 4, 303–328.

FREEMAN, L. 1979. Centrality in social networks conceptual clarification. *Social networks 1,* 3, 215–239.

FREEMAN, L. C. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry 40,* 1, 35–41.

GEORGE, B., KIM, S., AND SHEKHAR, S. 2007. Spatio-temporal network databases and routing algorithms: A summary of results. In *Symposium on Spatial and Temporal Databases (SSTD)*. 460–477.

GEORGE, B. AND SHEKHAR, S. 2006. Time-aggregated graphs for modeling spatio-temporal networks. *J. Data Semantics 11*, 191–212.

HABIBA, TANTIPATHANANANDH, C., AND Y. BERGER-WOLF, T. 2007. Betweenness centrality measure in dynamic networks. Tech. Rep. 2007-19, Center for Discrete Mathematics and Theoretical Computer Science.

HOWISON, J., WIGGINS, A., AND CROWSTON, K. 2011. Validity issues in the use of social network analysis with digital trace data. *Journal of the Association for Information Systems 12,* 12.

JOHNSON, R., KOVCS, B., AND VICSEK, A. 2012. A comparison of email networks and off-line social networks: A study of a medium-sized bank. *Social Networks*, –.

KARAGIANNIS, T. AND VOJNOVIC, M. 2008. Email information flow in large-scale enterprises. Technical Report Microsoft Research.

KIM, H. AND ANDERSON, R. 2012. Temporal node centrality in complex networks. *Physcial Review E 85,* 2.

KIM, H., TANG, J., ANDERSON, R., AND MASCOLO, C. 2012. Centrality prediction in dynamic human contact networks. *Computer Networks 56,* 3, 983 – 996.

KIM, P.-J. AND JEONG, H. 2007. Reliability of rank order in sampled networks. *The European Physical Journal B - Condensed Matter and Complex Systems 55,* 1, 109–114.

KLEINBERG, J. 2002. Bursty and hierarchical structure in streams. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '02. ACM, 91–101.

KOSSINETS, G., KLEINBERG, J., AND WATTS, D. 2008. The structure of information pathways in a social communication network. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '08. 435–443.

LERMAN, K. AND GHOSH, R. 2010. Information contagion: An empirical study of the spread of news on digg and twitter social networks. In *Proceedings of 4th International Conference on Weblogs and Social Media (ICWSM)*.

LERMAN, K., GHOSH, R., AND KANG, J.-H. 2010. Centrality metric for dynamic network analysis. In *Proceedings of KDD workshop on Mining and Learning with Graphs (MLG)*.

LESKOVEC, J., BACKSTROM, L., AND KLEINBERG, J. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. 497–506.

MALMGREN, R. D., HOFMAN, J. M., AMARAL, L. A., AND WATTS, D. J. 2009. Characterizing individual communication patterns. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. ACM, New York, NY, USA, 607–616.

MALMGREN, R. D., STOUFFER, D. B., MOTTER, A. E., AND AMARAL, L. A. N. 2008. A poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences*.

MYERS, S. A., ZHU, C., AND LESKOVEC, J. 2012. Information diffusion and external influence in networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '12. 33–41.

NIA, R., BIRD, C., DEVANBU, P., AND FILKOV, V. 2010. Validity of network analyses in open source projects. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR)*. 201–209.

SULO, R., BERGER-WOLF, T., AND GROSSMAN, R. 2010. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*. MLG '10. ACM, New York, NY, USA, 127–136.

TANG, J., MUSOLESI, M., MASCOLO, C., LATORA, V., AND NICOSIA, V. 2010. Analysing information flows and key mediators through temporal centrality metrics. In *Proceedings of the 3rd Workshop on Social Network Systems*. SNS '10. 3:1–3:6.

TANTIPATHANANANDH, C., BERGER-WOLF, T., AND KEMPE, D. 2007. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '07. 717–726.

TYLER, J. R. AND TANG, J. C. 2003. When can i expect an email response? a study of rhythms in email usage. In *Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*. 239–258.

WANG, D., WEN, Z., TONG, H., LIN, C.-Y., SONG, C., AND BARABSI, A.-L. 2011. Information spreading in context. In *Proceedings of the 20th international conference on World wide web*. WWW '11. ACM, 735?744.

WENG, L., FLAMMINI, A., VESPIGNANI, A., AND MENCZER, F. 2012. Competition among memes in a world with limited attention. *Scientific Reports 2*.

WU, F. AND HUBERMAN, B. A. 2007. Novelty and collective attention. *Proc. Natl. Acad. Sci. USA 104,* 45, 17599–17601.

YANG, J. AND LESKOVEC, J. 2010. Modeling information diffusion in implicit networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. IEEE Computer Society, 599–608.