Computation of Temporal Betweeness and Analysis of a Knowledge Mobilization Network *

Amir Afrasiabi Rad* Paola Flocchini* Joanne Gaudet†

Abstract

The temporal component of social networks is often neglected in their analysis, and statistical measures are usually performed on "static" network representations. As a result, measures of importance (like betweenness centrality) typically do not reveal the temporal role of the entities involved. Our goal is to contribute to fill this limitation by proposing a form of temporal betweenness measure (*foremost betweenness*) to analyse a knowledge mobilization network. We first describe a new algorithm to compute foremost betweenness, we then show that this measure, which takes time explicitly into account, allows us to detect centrality roles that were completely hidden in the classical statistical analysis. In particular, we uncover nodes whose static centrality was considered negligible, but whose temporal role is instead important to accelerate mobilization flow in the network. We also observe the reverse behaviour by detecting nodes with high static centrality, whose role as temporal bridges is instead very low. By revealing important temporal roles, this study is a first step towards a better understanding of the impact of time in social networks, and opens the road to further investigation.

Keywords. Time-varying graphs, temporal betweenness, dynamic networks, temporal analysis, social networks.

1 Introduction

Highly dynamic networks are networks where connectivity changes in time and connection patterns display possibly complex dynamics. Such networks are more and more pervasive in everyday life and the study of their properties is the object of extensive investigation in a wide range of very different contexts. Some of these contexts are typically studied

*School of Electrical Engineering and Computer Science, University of Ottawa, Canada. a.afrasiabi@uOttawa.ca

†School of Computer Science, Carleton University, Canada. flocchin@site.uottawa.ca

‡Alpen Path Solutions Inc. Ottawa, Canada. jgaudet@magma.ca

*A preliminary version of this paper appeared in the Proc. of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Workshop on Dynamics in Networks, 2015

in computer science, such as wireless, adhoc networks, transportation, vehicular networks, satellites, military and robotic networks (e.g., see [6, 7, 16, 23, 25, 26]); while others belong to totally different disciplines. This is the case for example, of the nervous system, livestock trade, epidemiological networks, and multiple forms of social networks (e.g., see [21, 24, 27, 28, 29, 31]). Clearly, while being different in many ways, these domains display common features; *time-varying graphs* (TVGs) represent a model that formalizes highly dynamic networks encompassing the above contexts into a unique framework, and emphasizes their temporal nature [8].

Knowledge Mobilization (KM) refers to the use of knowledge towards the achievement of goals [13]. Scientists, for example, use published papers to produce new knowledge in further publications to reach professional goals. In contrast, patient groups can use scientific knowledge to help foster change in patient practices, and corporations can use scientific knowledge to reach financial goals. Recently, researchers have started to analyse knowledge mobilization networks (KMN) using a social network analysis (SNA) approach (e.g., see [3, 5, 9, 10, 20]). In particular, [14] proposed a novel approach where a heterogeneous network composed of a main class of actors subdivided into three sub-types (individual human and non-human actors, organizational actors, and non-human mobilization actors) associated according to one relation, knowledge mobilization (a Mobilization-Network approach). Data covered a seven-year period with static networks for each year. The mobilization network was analysed using classical SNA measures (e.g., node centrality measures, path length, density) to produce understanding for KM using insights from network structure and actor roles [14].

The KM SNA studies mentioned above, however, lack a fundamental component: in fact, their analysis is based on a static representation of KM networks, incapable of sufficiently accounting for the time of appearance and disappearance of relations between actors beyond static longitudinal analysis. Indeed, incorporating the temporal component into analysis is a challenging task, but it is undoubtedly a critical one, because time is an essential feature of these networks. Temporal analysis of dynamic graphs is in fact an important and extensively studied area of research (e.g., see [12, 19, 17, 18, 30, 32, 33]), but there is still much to be discovered. In particular, most temporal studies simply consider network dynamics in successive static snapshots thus capturing only a partial temporal component by observing how static parameters evolve in time while the network changes. Moreover, very little work has been dedicated to empirically evaluating the usefulness of metrics in time (e.g., see [1, 22]).

In this paper, we represent KMN by TVGs and we propose to analyse them in a truly temporal setting. We design an algorithm to compute a form of temporal betweenness in time-varying graphs (*foremost betweenness*) that measures centrality of nodes in terms of how often they lie within temporal paths with earliest arrival. We then provide, for the first time on a real data set, an empirical indication of the effectiveness of foremost betweenness. In particular, we focus on data extracted from [14], here referred to as *Knowledge-Net*. We first consider static snapshots of Knowledge-Net corresponding to the seven years of its existence, and by studying the classical centrality measures in those time intervals, we provide rudimentary indications of the networks' temporal behaviour. To gain a finer temporal understanding, we then concentrate on *temporal betweenness* following a totally different approach. Instead of simply observing the static network over consecutive time intervals,

we focus on the TVG that represent Knowledge-Net and we compute foremost betweenness, explicitly and globally taking time into account. We compare the temporal results that we obtain with classical static betweenness measures to gain insights into the impact that time has on the network structure and actor roles. We notice that, while many actors maintain the same role in static and dynamic analysis, some display striking differences. In particular, we observe the emergence of important actors that remained invisible in static analysis, and we advance explanations for these. Results show that the form of temporal betweenness we apply is effective at highlighting the role of nodes whose importance has a temporal nature (e.g., nodes that contribute to mobilization acceleration). This research opens the road to the study of other temporal measures designed for TVGs.

2 Time-Varying Graphs

2.1 Definition

Time-varying Graphs are graphs whose structure varies over time. Following [8], a time-varying graph (TVG) is defined as a quintuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where V is a finite set of nodes; $E \subseteq V \times V$ is a finite set edges. The graph is considered within a finite time span $\mathcal{T} \subseteq \mathbb{T}$, called lifetime of the system. $\rho : E \times \mathcal{T} \rightarrow \{0, 1\}$ is the edge presence function, which indicates whether a given edge is available at a given time; $\zeta : E \times \mathcal{T} \rightarrow \mathbb{T}$, is the latency function, which indicates the time it takes to cross a given edge if starting at a given date. The model may, of course, be extended by defining the vertex presence function ($\psi : V \times \mathcal{T} \rightarrow \{0, 1\}$), and vertex latency function ($\phi : V \times \mathcal{T} \rightarrow \{0, 1\}$). The footprint of $\mathcal{G}$ is a static graph composed by the union of all nodes and edges ever appearing during the lifetime $\mathbb{T}$.

2.2 Journeys

A journey $\mathcal{J}$ in a TVG $\mathcal{G}$ is a temporal walk defined as a sequence of ordered pairs $\{(e_1, t_1), (e_2, t_2),...,(e_k, t_k)\}$, such that $\{e_1, e_2, ..., e_k\}$, called the journey route and represented by R, is a walk in G, if and only if $\rho(e_i, t_i) = 1$ and $t_{i+1} \geq t_i + \zeta(e_i, t_i)$ for all $i < k$. Every journey has a $departure(\mathcal{J})$ and an $arrival(\mathcal{J})$ that refer to journey's starting time t_1 and its last time $t_k + \zeta(e_k, t_k)$. Journeys are divided into three classes based on their variations based on the temporal and topological distance [4]. Journeys that have earliest arrival times are called *foremost* journeys, journeys with the smallest topological distance are referred to as *shortest* journeys, while the journey that takes the smallest amount of time is called *fastest*. Moreover, we call *foremost increasing journey* the ones whose route $\{e_1, e_2, \ldots, e_k\}$ is such that $\textit{birth-date}(e_i) \leq \textit{birth-date}(e_{i+1})$.

2.3 Temporal Betweenness

Betweenness is a classic measure of centrality extensively investigated in the context of social network analysis; the betweenness of a node $v \in V$ in a static graph $G = (V, E)$ is defined as follows:

$$B(v) = \sum_{u \neq w \neq v \in V} \frac{|P(u,w,v)|}{|P(u,w)|} \tag{1}$$

where $|P(u,w)|$ is the number of shortest paths from u to w in G, and $|P(u,w,v)|$ is the number of those passing through v. Even if static betweenness is "atemporal", we denote here by $B(v)^{\mathcal{T}}$ the static betweenness of a node v in a system whose lifetime is $\mathcal{T}$. Typically, vertices with high betweenness centrality direct a greater flow, and thus, have a high load placed on them, which is considered as an indicator for their importance as potential gatekeepers in the network.

While betweenness in static graphs is based on the notion of shortest path, its temporal version can be extended into three different measures to consider shortest, foremost, and fastest journeys for a given lifetime $\mathcal{T}$ [30].

In this paper we consider foremost betweenness. Nodes with a high foremost betweenness value do not simply act as gatekeepers of flow, like their static counter-part. In fact, they direct the flow that conveys a message in an *earliest* transmission fashion. In other words, intuitively, they provide some form of "acceleration" in the flow of information.

Foremost betweenness $TB_{\mathcal{F}}^{\mathcal{T}}(v)$ for node v with lifetime $\mathcal{T}$ is here defined as follows:

$$TB_{\mathcal{F}}^{\mathcal{T}}(v) = \frac{n(v)}{n} \sum_{u \neq w \neq v \in V} \frac{|\mathcal{F}^{\mathcal{T}}(u,w,v)|}{|\mathcal{F}^{\mathcal{T}}(u,w)|} \tag{2}$$

where $|\mathcal{F}^{\mathcal{T}}(u,w)|$ is the number of foremost *journey routes* between u and w during time frame $\mathcal{T}$ and $|\mathcal{F}^{\mathcal{T}}(u,w,v)|$ is the number of the ones passing through v in the same time frame, n is the total number of nodes, and $n(v)$ is the number of nodes in the connected component to which v belongs. The factor $\frac{n(v)}{n}$ is an adjustment coefficient to take into account possible network disconnections.

3 The Algorithms

3.1 Temporal Shortest Betweenness

Counting the shortest journeys in TVGs can be done employing the algorithm developed in [4] to construct a shortest journey spanning tree from a source to all destinations. In order to count the shortest journeys, slight modifications are needed, to store also the number of shortest journeys arriving at each vertex at each step of time and record the number of hops that they have had so far in the journey.

Algorithm 1 (CountShortest) contains this modification: the algorithm receives (G, s) as its input, where G is the TVG and s is the starting node from which the journeys to all other vertices in the TVG are being counted. The results are returned in the combination of $shortestCount[v,k]$ and $shortestIntCount[v,k]$ matrices, which record the number of shortest journeys from s to v with length k, and the number of such journeys that pass through the nodes that fall on the path of the corresponding journey.

The algorithm starts by adding all the possible predecessors of v into the predecessor list. Matrix $Pred[v,k]$ stores the predecessor of vertex v. Each predecessor falls on a "quasi-

Algorithm CountShortest
input : A TVG G, a vertex $s \in V_G$
output: $shortestCount[v, k]$: number and length of the shortest journeys from s to all $v \in V_G$

```
begin
    Initialize t_LBD[s] ← 0, Pred[s, 0] ← (), k ← 0, arr ← (),
    shortestCount[{., .}] ← 0, count[{., .}] ← 0, intCount[{., .}] ← ∅, and define for all
    v ≠ s, t_LBD[v] ← ∞
    while there is v ∈ V_G such that t_LBD(v) = ∞ and k < n do
        k ← k + 1
        arr ← ()
        for (u, v) ∈ V_G do
            Let t = EarliestTransmit((u, v), t_LBD[u])
            if (t + ζ(u, v)) ≤ t_LBD[v] then
                add (u, (t + ζ(u, v))) to Pred[v, k]
                arr[v] ← min((t + ζ(u, v)), arr[v])
            end
        end
        for (w, k − 1) ∈ Pred[v, k] do
            count[{v, k}] ← count[{v, k}] + count[{w, k − 1}]
            for each (x, i) in intCount[{w, k − 1}] do
                if x exists in intCount[{v, k}] as (x, j) then
                    replace (x, j) with (x, i + j) in intCount[{v, k}] and update the
                    count for (x, i + j)
                end
                else
                    add (x, i + 1) to the end of intCount[{v, k}] and update the count
                    for (x, i + 1)
                end
            end
        end
        for v ∈ V_G do
            if t_LBD[v] = ∞ then
                shortestCount[{v, k}] = count[{v, k}]
                update shorthestIntCount[{v, k}] with intCount[{v, k}]
                t_LBD[v] = arr[v]
            end
        end
        t_LBD ← arr
    end
end
```

Algorithm 1: An algorithm to count all shortest journeys from s to all nodes.

shortest path" with length k. We call this path quasi-shortest since it does not always store the shortest path, rather, it carries some longer paths that might contribute to the shortest path at later hops. The arrival time to some vertex v at the current step is stored in variable $arr(v)$, which is recorded for future path feasibility check. The quasi shortest paths are counted and stored in Matrix $count[v,k]$, while the $intCount[v,k]$ stores the number of journeys that pass through a specific vertex on the quasi shortest path from s to v. Some quasi-shortest paths are, indeed, the shortest paths that are recorded in $shortestCount[v,k]$, and the count for their intermediate vertices are stored in the $shortestIntCount[v,k]$. This is determined by checking the local array $t_{LBD}[u]$ that gives for each $u \in V_G$ a lower bound on the departure time, meaning the earliest time that the journey can exit u. $t_{LBD}[u]$ is initialized to infinity and gets updated anytime that a shortest journey to a vertex is found. Thus, checking whether the value of $t_{LBD}[u]$ for u is infinity or not, determines if the shortest journey for u is found earlier or not. It should be noted that function $EarliestTransmit(,)$ gives, for each edge (u,v), and each time instant t, the earliest moment after t when vertex u can transmit a message to v. If such a moment does not exist, $EarliestTransmit(,)$ returns $+\infty$.

Algorithm CountShortest counts the number of shortest journeys from one node to all the others, and also the number of such journeys that pass through each intermediate vertex. Repeating this procedure for all starting points would provide all the necessary information to compute temporal shortest betweenness for all nodes.

Let δ indicate the maximum number of different time intervals on an edge. The complexity analysis follows directly from the one of the original algorithm described in [4] and we have:

Theorem 3.1 *The computation of temporal shortest betweenness of all nodes in a TVG can be computed in* $O(n^4 \log \delta)$.

Proof Using the data structure proposed in [4], procedure $EarliestTransmit(,)$ is computed in time $O(\log \delta)$ due to the fact that we can apply binary search to find the earliest transmit time. We call procedure $EarliestTransmit(,)$, m times during the execution of the algorithm for a total of $O(m \log \delta)$. Meanwhile, $t_{LBD}[v], \forall v \in V_G$ becomes a value smaller than infinity if and only if the graph is connected and we have found all the shortest journeys including the longest one, which is equal to the eccentricity. In case of a disconnected TVG, we have to iterate the *while loop* at most n times. The other factor contributing to the complexity appears in the nested loop for that results which amounts to $O(n^2)$ times.

The complexity has to be multiplied by n to repeat the process from every possible starting node. ∎

3.2 Foremost Betweenness

The situation is rather different when computing foremost betweenness. In fact, it is easy to see that there exist TVGs where counting all foremost journeys or journey routes between two vertices is #P-complete, which means that no polynomial can be devised.

Consider, for example, TVGs where edges always exist (note that a static graph is a particular TVG) and latency is zero. In such a case any journey between any pair of nodes

is a foremost journey. Counting all of them is then equivalent to counting all paths between them, which is a #P-complete problem (see [34]). In general, it is then unavoidable to have a worst case exponential algorithms to compute foremost betweenness in an arbitrary TVG.

In this Section we first focus on foremost betweenness based on journey routes in the general setting (Algorithm 2). We then focus on foremost betweenness for special TVGs with zero latency and instant edges (Algorithm 3), which correspond to the characteristics of the knowledge Mobilization Network that we analyze in Section 4. Note that each solution has the same worst case time complexity, linear in the total number of journey routes in the TVG, which can clearly be exponential. The advantages of the algorithm designed for the special temporal condition of instant edges and zero latency are mainly practical. In fact, the worst case complexities are the same, but the execution time can be significantly better in practice.

3.2.1 A General Algorithm

In this Section we describe an algorithm for the computation of all journey routes between a node to all other nodes, which is at the basis of the computation of foremost betweenness.

The input of Algorithm CountFormemostJourney is a pair (G, s) where $G = (V, E)$ is a TVG and s a starting node; the algorithm returns a matrix $Count_s[x, y]$, for all $x, y \in V$ containing the number of foremost journeys from s to y passing through x (note that $Count_s[x, x]$ denotes the number of foremost journeys from s to x).

First of all, the earliest arrival times of foremost journeys starting from s to all nodes are computed using the Algorithm from [4]. To each node v is then associated its foremost arrival time $foremost(v)$.

The counting algorithm is simple and it is based on Depth-First Search (DFS) traversal. It essentially consists of visiting every journey route of G starting from s, incrementing the appropriate counters every time a newly encountered journey is foremost. A typical DFS traversal visits every node and terminates when they are all visited; in our algorithm, however, we need to repeatedly perform DFS, re-visiting nodes possibly several times, so to traverse all journey routes.

To do so, the traversal starts as a usual DFS, pushing the incident edges of the source s onto a stack S and visiting one of the adjacent neighbours (say s'), thus discovering a first journey $\pi_0 = [(s, s')]$. The current journey under visit is kept in a second stack $Path$ (nodes in $Path$ are marked *visited*). At this point the DFS continues pushing on the stack the edges incident to s' that are feasible with π_0 (i.e., the edges whose latest traversal time is greater than or equal to the earliest arrival time at s'), and updating $Count[s', s']$ if π_0 has a foremost arrival time at s'.

In general, as soon as a journey $\pi = [(x_0, x_1), (x_1, x_2), \ldots, , (x_{k-1}, x_k)]$ is encountered in the traversal, $Count[x_i, x_k]$, $i \leq k$ is updated only if π is a foremost journey, and, regardless of it being foremost, the traversal continues pushing on the stack the edges incident to x_k that are temporally feasible with π.

Whenever backtracking is performed, however, the already visited nodes on the backtracking path are remarked *unvisited* (and popped from $Path$) in such a way that they can be revisited as part of different journey route, not yet explored. We remind that in a journey route a node can re-appear more than once, with the various occurrences corresponding to

different times. We then need to store the moment when a node is visited in the journey route so that, if it is visited again, we can determine whether the second visit happened at a later time. Thus, in both $Path$ and S stacks, we store also a time-stamp, to register the time of the first visit in $Path$ and to register the time for the next visits in the S. If the visits happens at different times, the same node is pushed into the stacks again. In fact, we push the nodes to the $Path$ or S stacks only if they are not visited yet (i.e., not in the path), or they are visited before (in the path) but at a different time. Function $arriv(x, y, t)$ returns the arrival time to y, leaving x at time t.

Algorithm CountFormemostJRoutes**.**
input : (G, s) : a TVG $G = (V, E)$, $s \in V$
output: $Count_s[x, y]$, $\forall x, y \in V$: number of foremost journey routes from s to $y \in V$, passing through $x \in V$
begin
  $Path.push(s, 0), Count_s[., .] \leftarrow 0$
  for *all* $w \in Adj(s)$ **do**
    $S.push(s, w, arriv(s, w, 0))$
  end
  while $S \neq \emptyset$ **do**
    $(x, y, t) \leftarrow S.pop()$
    while $x \neq Path.top()$ **do**
      $Path.pop()$
    end
    Let π be the journey route corresponding to the content of $Path$
    Let $t_{x,y}$ be the latest possible traversing time of edge (x, y)
    if $t_{x,y} \geq arrival(\pi)$ **then**
      if $y \notin Path$ *or* $y \in Path$ *at time* $t' < t$ **then**
        $Path.push(y, arriv(x, y, t))$
        for *each* (y, w) *such that* $t_{y,w} \geq arrival(\pi)$ *and either* $w \notin Path$ *or* $w \in Path$ *at time* $t' < arriv(y, w, t)$ **do**
          $S.push(y, w, arrival(y, w, t))$
        end
        if $arrival(\pi) = foremost(y)$ **then**
          Update $Count_s[z, y]$ for all $z \in Path$
        end
      end
    end
  end
end

Algorithm 2: Algorithm to count all journey routes from s to all the other nodes.

Let μ be the number of different journey routes in G, and $\mathcal{N}(\mu)$ the number of nodes on those routes.

Theorem 3.2 *Algorithm 2 counts all journey routes from a source to all the other nodes in G in $O(\mathcal{N}(\mu))$ time.*

Proof Correctness of the algorithm is straightforward as it is easy to see that the algorithm follows multiple DFS traversals to visit every journey route from a given source, counting the number of such routes from s to any destination passing through any intermediate node.

The algorithm traverses every journey from s to any other node and it performs an update for every visited node in each foremost journey that it encounters. Thus, in the worst case, it has a $O(\mathcal{N}(\mu))$ time complexity.

Clearly, to compute foremost betweenness based on journey routes, the algorithm has to be repeated for every possible source for a total complexity of of $O(n\mathcal{N}(\mu))$.

3.2.2 Algorithm for Zero latency and Instant edges

Algorithm 2 is applicable to a general TVG. We now consider a very special type of TVG with specific temporal restrictions that correspond to the type of network that we analyze in the next section. One such peculiarity is given by *instant edges*, i.e., edges that appear only during a unique time interval, another characteristic is zero latency: an edge can be traversed instantaneously.

We now describe a variation of the algorithm specifically designed for those conditions and we compute foremost betweenness based on increasing journey routes, i.e., we apply the foremost betweenness formula using foremost increasing journeys.

Given a TVG $G = (V, E)$, since we assume the presence of instant edges, we can divide time in consecutive intervals $I_1, I_2, \ldots, I_k$ corresponding to k snapshots $G_1, G_2, \ldots G_k$ ($G_i = (V_i, E_i)$), in such a way that $(x, y) \in E_i$ implies that $(x, y) \notin E_j$ for $j \neq i$. Furthermore, we know that $\zeta = 0$, that is an edge can be traversed in zero time.

The key idea that can be applied to this very special structure is based on the observation that, given a foremost route $\pi_{x,y}$ from x to y with edges in time intervals I_j, with $j > i$, and given any journey route $\pi'_{s,x}$ from s to x with edges only in I_i, the concatenation of $\pi'_{s,x}$ and $\pi_{x,y}$ is a foremost route from s to y, passing through x.

This observation leads to the design of an algorithm that starts by counting the foremost routes belonging to the last snapshot G_k only, and proceeds backwards using the information already computed. More precisely, when considering snapshot G_i from a source s, the goal is to count all foremost routes involving only edges in $\cup_{j \geq i} E_j$ (i.e., with time intervals in $\cup_{j \geq i} I_j$), and when doing so, all the foremost routes involving only edges strictly in the "future" (i.e., time intervals $\cup_{j > i} I_j$) have been already calculated for any pair of nodes. The already computed information is used when processing snapshot G_i avoid a recalculation in a dynamic programming fashion.

The inputs of Algorithm 3 are: a snapshot G_i and a starting node s. The algorithm returns an array of lists, $Count_s[u, v]$, where each of the list elements refer to vertices falling on the journey. $Count_s[u, v]$, for all $u, v \in V$ contains the number of foremost journeys from s to u passing through v counted so far (i.e., considering only edges in $\cup_{j \geq i} E_j$).

The actual counting algorithm on snapshot G_i is a modified version of Algorithm 2, still based on Depth-First Search (DFS) traversal. However, when a new route is discovered to

```
Algorithm CountAllSpecial
input  : A TVG G_i, starting node s ∈ V, and snapshot interval I
output: Count_s[v, u] that records the number of the journeys from s ∈ V_G to all
        u ∈ V_G passing through v ∈ V_G during interval I
begin
    Initialize Count_s[., .] ← 0
    Path.push(s)
    for all w ∈ Adj(s) do
        S.push(s, w)
    end
    while S ≠ ∅ do
        (x, y) ← S.pop()
        while x ≠ Path.top() do
            Path.pop()
        end
        if y ∉ Path then
            Path.push(y)
            if y falls in snapshot interval I then
                for each (y, w) such that w ∉ Path do
                    S.push(y, w)
                end
                if path is foremost then
                    Count_s[z, y] = Normal Count  Count_s[z, y] for all z ∈ Path
                end
            end
            else
                Count_s[z, y] = Special Count  Count_s[z, y] for all z ∈ Path
            end
        end
    end
end
```

Algorithm 3: Counting all foremost journeys in TVGs with zero latency and instant edges.

some node x, if this route is foremost, a normal update is performed like in Algorithm 2: i.e., an increment to $Count_s[v, x]$ is done, v being the node that falls on the journey route from s to x. If instead it is not a foremost route and it is connected to a node that existed in the "future", a special update is performed using the data already calculated for the "future snapshots". In other words, when $s \rightsquigarrow x$ is a prefix of a journey route $x \rightsquigarrow y$ at a later time snapshot, we perform a procedure called SpecialCount. The special count involves aggregating the values of $Count_s[v, x]$ with $Count_x[v', y]$, for all nodes occurring in the journey routes between s and x and between x and y (see Algorithm 4).

The time complexity of Algorithm 3, CountAllSpecial, is the same as the one of the general algorithm, CountFormemostJRoutes. However, in this particular case, the size

Procedure SpecialCount.
input : $Count_s[.,x]$, in G_i, and $Count_x[.,y]$ in $\cup_{j>i} G_j$
output: $Count_s[v,y]$, $\forall v \in V$: number of foremost journey routes from s to $y \in V$, passing through $v \in V$
begin
  for *each* $v \in U \cup W$ *where* $U =$ *all nodes in* $x \rightsquigarrow y$ *and* $W =$ *all nodes in* $s \rightsquigarrow x$ **do**
    if $v \in s \rightsquigarrow x$ *and* $v \notin x \rightsquigarrow y$ **then**
      $Count_s[v,y] += Count_s[v,x] \times Count_x[y,y]$
    end
    else if $v \notin s \rightsquigarrow x$ *and* $v \in x \rightsquigarrow y$ **then**
      $Count_s[v,y] += Count_s[x,x] \times Count_x[v,y]$
    end
    else if $v \in s \rightsquigarrow x$ *and* $v \in x \rightsquigarrow y$ **then**
      $Count_s[v,y] += Count_s[x,x] \times Count_x[v,y] + Count_s[v,x] \times Count_x[y,y] - Count_s[v,x] \times Count_x[v,y]$
    end
  end
end

Algorithm 4: The SpecialCount module.

of a journey route can be bounded by nk, where k is the number of snapshots of $\mathcal{G}$.

– **Practical Considerations: reducing time.** Algorithm 3 has to be executed in the chronological order of the time corresponding to the different snapshots, starting from the last one, since it uses the previously calculated results in the computation of the new results. Since the graph is divided into independent snapshots, the number of all journeys can be computed separately for each snapshot, and the result of the calculation can be aggregated at the end. This has the advantage of reducing the time complexity of the computation eliminating all the special updates from the first part of the algorithm (while detecting all the journey routes) and deferring it to the second part (when aggregating all the information for the final update). Thus, instead of performing the special count at each level, we can postpone it to the last step of the algorithm, and loop once through all the collected counts with hard-coded intervals in the loop. While not being advantageous in worst case scenarios, this strategy results in a generally much more efficient solution from a practical point of view.

4 Knowledge-Net

Knowledge-Net is an heterogeneous network where nodes represent human and non-human actors (researchers, projects, conference venues, papers, presentations, laboratories), and edges represent knowledge mobilization between two actors. The network was collected for a period of seven years [14]. Once an entity or a connection is created, it remains in the system for the for entire period of the analysis.

Table 1 provides a description of the *Knowledge-Net* dataset. The dataset consists of

Table 1: *Knowledge-Net* data set with characteristics of actors and their roles at different times

Start	Duration	#Nodes	#Edges	Granularity
2005	7 Years	366	750	1 Year

Actor Type	2005	2006	2007	2008	2009	2010	2011
HIA	3	22	27	46	51	76	94
NHIA	0	3	6	9	9	9	15
NHMA	7	25	43	87	132	194	248
OA	0	5	5	9	9	9	2
Total	10	55	81	151	201	288	366

366 vertices and 750 edges in 2011. The number of entities and connections vary over times starting from only 10 vertices and 14 edges in 2005 and accumulating to the final network year in 2011. *Knowledge-Net* is mainly comprised of non-human actors, 272 in total (non-human mobilization actors, NHMA, non-human individual actors, NHIA, and organizational actors, OA), in relation with 94 human actors (HA). Human actors include principle investigators (PI), highly qualified personnel (HQP) and collaborators (CO). It is through mobilization actors (NHMA) that individual, organizational actors and mobilization actors associate and mobilize knowledge to reach goals. For example, scientists mobilize knowledge through articles where not all contributing authors might be in relation with all other authors, yet all relate with the publication [14]. These non-human mobilization actors make up the bulk of the network including conference venues, presentations (invited oral, non-invited oral and poster), articles, journals, laboratories, research projects, websites, and theses.

Classical statistical parameters have been calculated for Knowledge-Net, representing it as a static graph where the time of appearance of nodes and edges did not hold any particular meaning. In doing so, several interesting observations were made regarding the centrality of certain nodes as knowledge mobilizers and the presence of communities [14]. In particular, all actor types increased in number over the 7 years indicating a rise in new mobilization relations over time. Although non-human individual actor absolute numbers remained small (ranging from 3 in 2006 to 15 in 2011), these actors were critical to making visible tacit (non-codified) knowledge mobilization from around the world (mostly laboratory material sharing, including from organizations and universities in the USA, from Norway, and from Canadian universities). Finally, embedded in human individual actor counts were individuals that the laboratory acknowledged in peer-reviewed papers, thus making further tacit and explicit knowledge mobilization visible.

When representing *Knowledge-Net* as a TVG, we notice that the latency ζ is always zero, as an edge represents a relationship and its creation does not involve any delay; moreover,

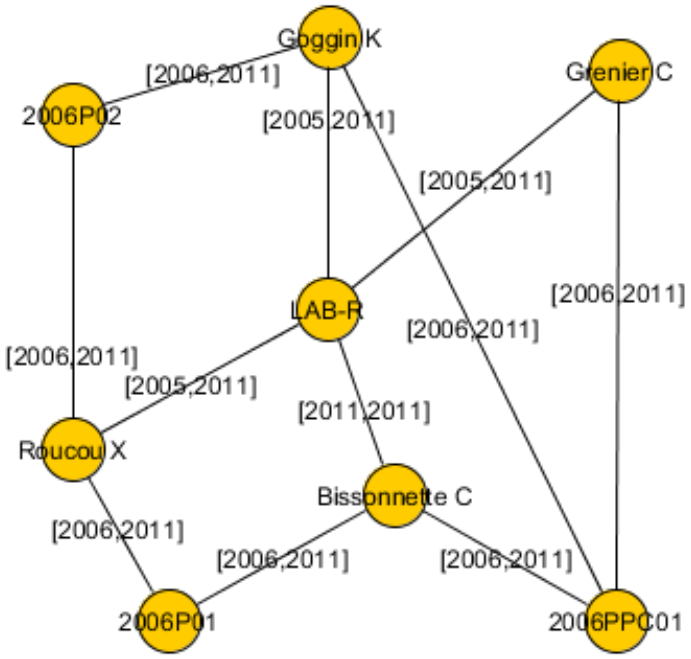

Figure 1: A small portion of Knowledge-Net represented as a TVG.

edges and nodes exist from their creation (their birth-date) to the end of the system lifetime. Let *birth-date*(e) denote the year when edge e is created. An example of a small portion of *Knowledge-Net* represented as a TVG is given in Figure 1.

We also notice that, due to zero latency and to the fact that edges never disappear once created, any shortest journey route in $\mathcal{G}$ is equivalent to a shortest path on the static graph corresponding to its footprint; moreover, the notion of fastest journey does not have much meaning in this context, because on any route corresponding to a journey, there would be a fastest one. On the other hand, the notion of foremost journey, and in particular of foremost increasing journey, is extremely relevant as it describes timely mobilization flow, i.e., flow that arrives at a node as early as possible.

Note that in this setting, the computation of foremost betweenness can be performed using Algorithm 3 introduced in the previous Section.

5 Study of KnowledgeNet

5.1 Preliminary Analysis on Consecutive Snapshots

To provide more clear statistics on the Knowledge-Net dataset and a ground for better understanding of temporal metrics, we first calculated classical statistical measures (e.g., node centrality measures, path length, density) on the seven static graphs, corresponding to the seven years of study. The average for each value for the graphs is calculated to represent a benchmark on how the rank for each node is compared to others.

The statistical data presented in Table 2 provides valuable information about the graph.

Table 2: Some static statistical parameters calculated for successive snapshots

	2005	2006	2007	2008	2009	2010	2011
Ave. Degree	1.40	1.32	1.63	1.84	1.98	2.02	2.04
Diameter	4	5	5	6	6	6	6
Density	0.31	0.04	0.04	0.02	0.02	0.01	0.01
#Communities	4	3	6	8	8	15	12
Modularity	0.17	0.52	0.46	0.47	0.46	0.54	0.54
Ave. Clustering Coefficient	0.41	0.06	0.21	0.22	0.20	0.24	0.23
Ave. Path Length	2.04	3.04	3.06	3.26	3.34	3.46	3.50
Ave. Normalized Closeness	0.51	0.33	0.33	0.31	0.30	0.29	0.29
Ave. Eccentricity	3.10	4.41	4.40	4.70	4.80	4.83	4.83
Ave. Betweenness	4.70	58.36	83.53	169.70	234.89	354.23	456.18
Ave. Normalized Betweenness	0.13	0.03	0.02	0.01	0.01	≈ 0	≈ 0
Ave. Page Rank	0.10	0.01	0.01	≈ 0	≈ 0	≈ 0	≈ 0
Ave. Eigenvector	0.52	0.19	0.15	0.10	0.09	0.07	0.05

The steady decrease in the (normalized) centrality values confirms that the network growth is not symmetric, so the centrality values have long tails. The low value of normalized betweenness, along with the low values for density, confirms that the graph is coupled in a way that there are a great number of shortest paths between any two arbitrary vertices in the graph. This caused the betweenness for most vertices to be similar and quite low when compared to the ones of nodes with the highest betweenness. Low average path length is a sign that the network presents small world characteristics and the knowledge mobilization to the whole network is expected to be conducted only in a few hops. Meanwhile, the decreasing graph density along with the increasing average degree represent the slow growth in the number of edges compared to the number of nodes. Escalation in the number of communities with increase in graph modularity metrics shows that the knowledge mobilization actors tend to form communities as time progresses. As the normalized average betweenness decreases steadily, it can be concluded that a few vertices at each community play the role of mediators and create the link between communities.

Apart from these general observations, a static analysis of consecutive snapshots, does not provide deep temporal understanding. For example, it does not reflect which entities engage in knowledge mobilization in a timely fashion, e.g. by facilitating fast mobilization, or slowing mobilization flow.

To tackle some of these questions, we represent *Knowledge-Net* as a TVG and we propose to study it by employing a form of temporal betweenness that makes use of time in an explicit manner.

5.2 Foremost Betweenness of Knowledge-Net

In this Section we focus on *Knowledge-Net*, and we study $TB_{\mathcal{F}}^{\mathcal{T}}(v)$ for all v. Nodes are ranked according to their betweenness values and their ranks are compared with the ones obtained calculating their static betweenness $B^{\mathcal{T}}(v)$ in the same time frame. Given the different meaning of those two measures, we expect to see the emergence of different behaviours, and, in particular, we hope to be able to detect nodes with important temporal roles that were left undetected in the static analysis.

5.2.1 Foremost Betweenness during the lifetime of the system

Table 3 shows the temporally ranked actors accompanied by their static ranks, and the high ranked static actors with their temporal ranks, both with lifetime $\mathcal{T}$ = [2005-2011]. In our naming convention, an actor named $Xi(yy)$ is of type X, birth date yy and it is indexed by i; types are abbreviated as follows: H (human), L (Lab), A (article), C (conference), J (journal), P (project), C (paper citing a publication), I (invited oral presentation), O (oral presentation). Note that only the nodes whose betweenness has a significant value are considered, in fact betweenness values tend to lose their importance, especially when the differences in the values of two consecutive ranks are very small [11].

Interestingly, the four highest ranked nodes are the same under both measures; in particular, the highest ranked node (L1(05)) corresponds to the main laboratory where the data is collected and it is clearly the most important actor in the network whether considered in a temporal or in a static way. On the other hand, the table reveals several differences worth exploring. From a first look we see that, while the vertices highest ranked statically appear also among the highest ranked temporal ones, there are some nodes with insignificant static betweenness, whose temporal betweenness is extremely high. This is the case, for example, of nodes S1(10) and J1(06).

The case of node S1(10). To provide some interpretation for this behaviour we observe vertex S1(10) in more details. This vertex corresponds to a poster presentation at a conference in 2010. We explore two insights. First, although S1(10) has a relatively low degree, it has a great variety of temporal connections. Only three out of ten incident edges of S1(10) are connected to actors that are born on and after 2010, and the rest of the neighbours appear in different times, accounting for at least one neighbour appearing each year for which the data is collected. This helps the node to operate as a temporal bridge between different time instances and to perhaps act as a knowledge mobilization accelerator.

Second, S1(10) is close to the centre of the only static community present in [2010-2011] and it is connected to the two most important vertices in the network. The existence of a single dense community, and the proximity to two most productive vertices can explain its negligible static centrality value: while still connecting various vertices S1(10) is not the shortest connector and its betweenness value is thus low. However, a closer temporal look reveals that it plays an important role as an interaction bridge between all the actors that appear in 2010 and later, and the ones that appear earlier than 2010. This role remained invisible in static analysis, and only emerges when we pay attention to the time of appearance of vertices and edges. On the basis of these observations, we can interpret S1(10)'s high

Table 3: List of highest ranked actors according to temporal (resp. static) betweenness, accompanied by the corresponding static (resp. temporal) rank in lifetime [2005-2011].

Temporal to Static				Static to Temporal		
Actor	Temporal Rank	Static Rank		Actor	Static Rank	Temporal Rank
L1(05)	1	1		L1(05)	1	1
H1(05)	2	2		H1(05)	2	2
A1(06)	3	3		A1(06)	3	3
A2(08)	4	4		A2(08)	4	4
P1(06)	5	8		A5(08)	5	12
A3(07)	6	9		A4(09)	6	7
A4(09)	7	6		P2(08)	7	9
S1(10)	8	115		P1(06)	8	5
P2(08)	9	7		A3(07)	9	6
J1(06)	10	160		P3(10)	10	17
C1(07)	11	223		A6(11)	11	18
A5(08)	12	5		A8(09)	12	36
I1(09)	13	28		P4(10)	13	22
O1(05)	14	45		P5(11)	14	27
S2(05)	15	46		H2(05)	15	44
I2(05)	16	47		A7(09)	16	21
P3(10)	17	10		A9(10)	17	31
A6(11)	18	11		P5(11)	18	69
C2(10)	19	133		P6(10)	19	23
J2(09)	20	182				
A7(09)	21	16				

temporal betweenness value as providing a fast bridge from vertices created earlier and those appearing later in time. This lends support to the importance of poster presentations that can blend tacit and explicit knowledge mobilization in human - poster presentation - human relations during conferences and continue into future mobilization with new non-human actors as was the case for S1(10) [2].

The case of node J1(06). J1(06), the *Journal of Neurochemistry*, behaves similarly to S1(10) with its high temporal and low static rank. As opposed to S1(10), this node is introduced very early in the network (2006); however, it is only active (i.e. has new incident edges) in 2006 and 2007. It has only three neighbours, A1(06), A3(07), and C1(07), all highly ranked vertices statically (A1(06), A3(07)), or temporally (C1(07)). Since its neighbouring vertices are directly connected to each other or in close proximity of two hops, J1(06) fails to act as a static short bridge among graph entities. However, its early introduction and proximity to the most prominent knowledge mobilizers helps it become an important temporal player in the network. This is because temporal journeys overlook geodesic distances and are instead concerned with temporal distances for vertices. These observations might explain the high temporal rank of J1(06) in the knowledge mobilization network.

5.2.2 A Finer look at foremost betweenness

A key question is whether the birth-date of a node is an important factor influencing its temporal betweenness. To gain insights, we conducted a finer temporal analysis by considering $TB_{\mathcal{F}}^{\mathcal{T}}$ for all possible birth-dates, i.e, for $\mathcal{T} = [\text{x},2011]$, $\forall x \in \{2005, 2006, 2007, 2008, 2009, 2010, 2011\}$. This allowed us to observe how temporal betweenness varies depending on the considered birth-date.

Before concentrating on selected vertices (statically or temporally important with at least one interval), and analysing them in more detail, we briefly describe a temporal community detection mechanism that we employ in analysis.

Detection of temporal communities. We approximately detect communities existing in temporal networks. To detect communities involving x, we first determine the temporal foremost journeys arriving at or leaving from x. We then replace each journey with a single edge, creating a static graph with an edge between x and all the vertices that are reachable from or can reach x in a foremost manner. For instance, Fig. 2 shows the transformation of a graph into a directed weighted graph that is used for community detection. We finally apply existing directed weighted community detection algorithms to compute communities around x [15]. The model is an approximation since it overlooks the role that is played in communities by vertices that fall along journeys while not being their start or end-points; however, it is sufficient for our purposes to give an indication of the community formation around a node.

The case of node P1(06). This is a research project led by the principle investigator at L1(05). The project was launched in 2006 and its official institutional and funded elements wrapped-up in 2011. Data in Table 3 support that P1(06) has similar temporal and static

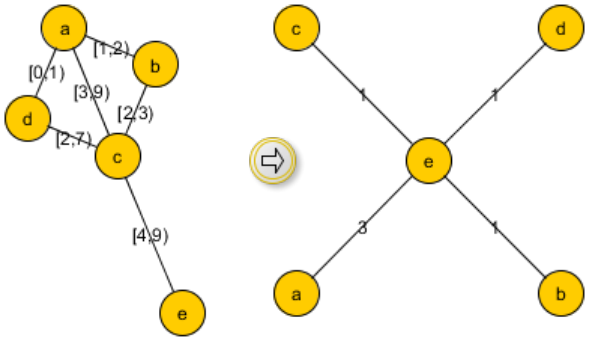

Figure 2: Transformation of a temporal graph into a weighted graph used for community detection.

ranks with regards to its betweenness in lifetime [2005-2011]. One could conclude that the temporal element does not provide additional information on its importance and that the edges that are incident to P(06)-1 convey the same temporal and static flow. However, there is still an unanswered question on whether or not edges act similarly if we start observing the system at different times. Will a vertex keep its importance throughout the system's lifetime?

The result of such analysis is provided in Fig. 3, where $TB_{\mathcal{F}}^{\mathcal{T}}(P1(06))$ is calculated for each birth-date (indicated in the horizontal axis), with all intervals ending in 2011.

While both equally important during the entire lifetime [2005-2011] of the study, this project seems to assume a rather more relevant temporal role when observing the system in a lifetime starting in year 2007 (i.e., $\mathcal{T}$ =[2007-2011]), when its static betweenness is instead negligible. This seems to indicate that the temporal flow of edges incident to P1(06) appearing from 2007 on is more significant than the flow of the edges that appeared previously.

With further analysis of P1(06)'s neighbourhood in [2007-2011], we can formulate technical explanations for this behaviour. First, its direct neighbours also have better temporal betweenness than static betweenness. Moreover, its neighbours belong to various communities, both temporally and statically. However, looking at the graph statically, we see several additional shortest paths that do not pass through P1(06) (thus making it less important in connecting those communities). In contrast, looking at the graph temporally P1(06) acts as a mediator and accelerator between communities. More specifically, we observe that the connections P1(06) creates in 2006 contribute to the merge of different communities that appear only in 2007 and later. When observing within interval [2006-2011], we then see that P1(06) is quite central from a static point of view, because the appearance of time of edges does not matter but, when observing it in lifetime [2007-2011] node P1(06) loses this role and becomes statically peripheral because the newer connections relay information in an efficient temporal manner.

In other words, it seems that P1(06) has an important role for knowledge acceleration in the period 2007-2011, a role that was hidden in the static analysis and that does not emerge even from an analysis of consecutive static snapshots. For research funders, revealing a research project's potentially invisible mobilization capacity is relevant. Research projects can thus be understood beyond mobilization outputs and more in terms of networked temporal

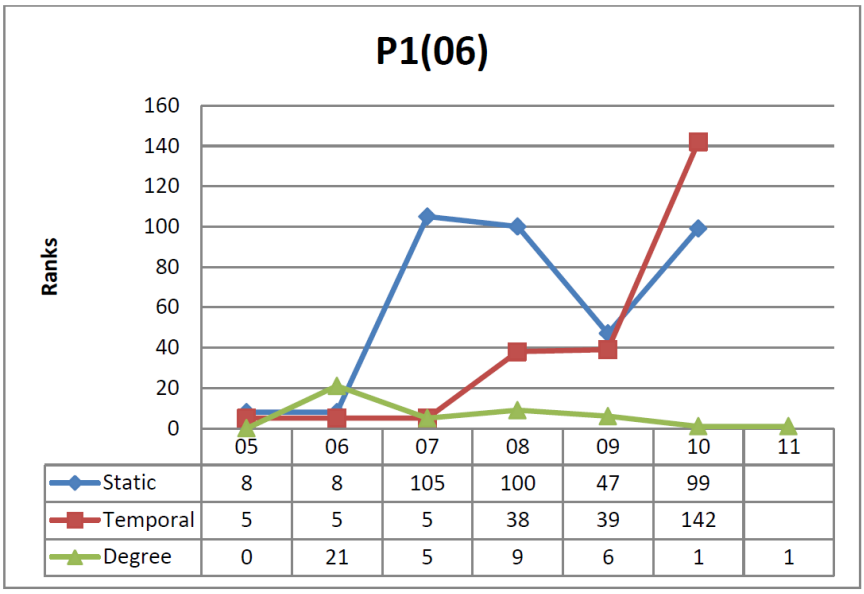

Figure 3: Comparison between different values for vertex P1(06). Ranks of the vertex in the last interval are not provided as both betweenness values are zero.

bridges to broader impact.

The case of node A3(07). The conditions for A3(07), a paper published in 2007, illustrate a different temporal phenomenon. Node A3(07) has several incident edges in 2007 (similarly to node P1(06)) when both betweenness measures are high. Peering deeper into the temporal communities formed around A3(07) is revealing: up to 2007, this vertex is two steps from vertices that connect two different communities in the static graph. The situation radically changes however with the arrival of edges in 2008 that modify the structure of those communities and push A3(07) to the periphery. The shift is dramatic from a temporal perspective because A3(07) loses it accelerator role where its temporal betweenness becomes negligible, while statically there is only a slight decrease in betweenness. The reason for a dampened decrease in static betweenness is that this vertex is close to the centre of the static community, connecting peripheral vertices to the most central nodes of the network (such as L1(05) and H1(05)). It is mainly proximity to these important vertices that sustains A3(07)'s static centrality.

Such temporal insights lend further support to understanding mobilization through a network lens coupled with sensitivity to time. A temporal shift to the periphery for an actor translates into decreased potential for sustained mobilization.

5.3 Invisible Rapids and Brooks

On the basis of our observations, we define two concepts to differentiate the static and temporal flow of vertices in Knowledge Mobilization networks. We call *rapids* the nodes with high foremost betweenness, meaning that they can potentially mobilize knowledge in a timelier manner; and *brooks* the ones with insignificant foremost betweenness. Moreover,

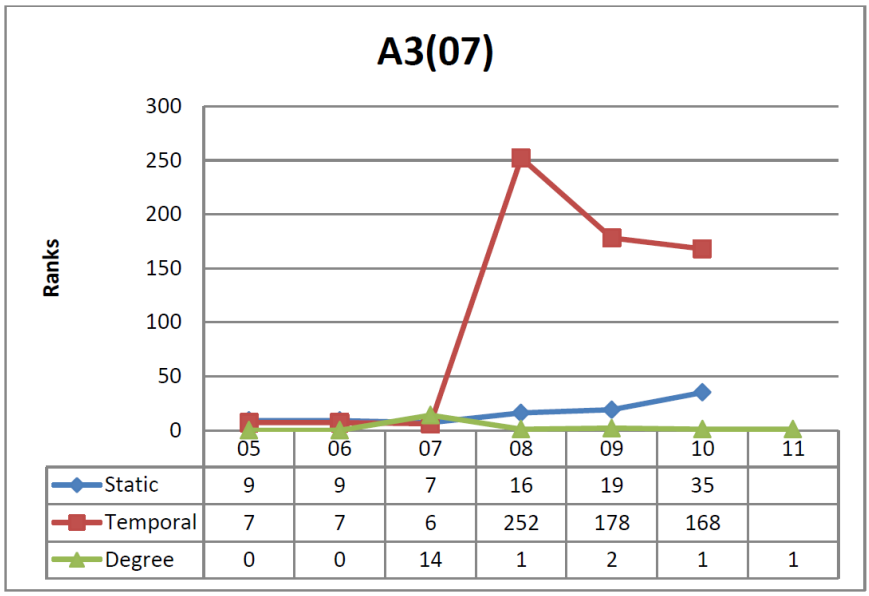

Figure 4: Comparison between different values for vertex A3(07). Ranks of the vertex in the last interval are not provided as both betweenness values are zero.

we call *invisible rapids* those vertices whose temporal betweenness rank is considerably more significant than their static rank (i.e., the ones whose centrality was undetected by static betweenness), and *invisible brooks* the ones whose static betweenness is considerably higher than their temporal betweenness, meaning that these vertices can potentially be effective knowledge mobilizers, yet they are not acting as effectively as others due to slow or non-timely relations.

Invisible rapids and brooks can be present in different lifetimes as their temporal role might be restricted to some time intervals only; for example, as we have seen in the previous Section, S1(10) and J1(06) are invisible rapids in $\mathcal{T}$ = [2005-2011], P1(06) is an invisible rapid in $\mathcal{T}$ = [2007-2011], A3(07) is an invisible brook in $\mathcal{T}$ = [2008-2011]. Tables 4 and 5 indicate the major invisible rapids and brooks observed in *Knowledge-Net*.

The presence of a poster presentation, a research project, two journals and a conference publication among the invisible rapids supports that different types of mobilization actors can impact timely mobilization while not being as effective at creating short paths among entities for knowledge mobilization. In other words, they can play a role of accelerating knowledge mobilization, but to a concentrated group of actors.

As for invisible brooks, we observe a journal (the *Biochemica et Biophysica Acta-Molecular Cell Research* (J3(08)), three papers (C3(11), C4(07), and C5(07)) that cite publications by the main laboratory in the study (L1(05)), a publication (A3(07)) mobilizing knowledge from members of L1(05), and a research assistant who worked on several research projects as an HQP. In comparison with invisible rapids, there is a wider variety in the type of mobilization actors that act as brooks which does not readily lend itself to generalization.

Interestingly, we see the presence of journals among invisible rapids and brooks. From our analysis, it seems that journals can hold strikingly opposite roles: on the one hand they can contribute considerably to more timely mobilization of knowledge while not being very

Table 4: Major invisible rapids

Actor	Time	Temp. Rank	Stat. Rank	Type
P1(06)	[07-11]	5	105	project
S1(10)	[05-11]	8	115	poster
	[06-11]	8	113	
	[07-11]	7	115	
	[08-11]	5	104	
J1(06)	[05-11]	10	160	journal
	[06-11]	10	154	
	[07-11]	10	223	
C1(07)	[05-11]	11	223	citing publication
	[06-11]	11	220	
J2(09)	[06-11]	17	179	journal
	[07-11]	16	182	
C2(10)	[05-11]	19	133	citing poster
	[06-11]	16	132	
	[07-11]	15	133	

strong bridges between communities; while on the other hand, they can play critical roles in bridging network communities, but at a slow pace. A brook, the journal *Biochemica et Biophysica Acta-Molecular Cell Research* (J3(08)), for example, helped mobilize knowledge in two papers for L1(05) (in 2008 and 2009) and is a journal in which a paper (in 2011) citing a L1(05) publication was also published. Given expected variability in potential mobilization for a journal, it is not surprising to see these mobilization actors at both ends of the spectrum.

In contrast, the presence of a research project as an invisible rapid is meaningful. It is meaningful in two ways. First, because when public funders invest in research projects as mobilization actor, an implicit if not explicit measure of success is timely mobilization with potential impact inside and outside of academia [14]. Ranking as a rapid (for a mobilization actor) is one measure that could therefore help funding agencies monitor and detect temporal change in mobilization networks. Second, a research project as rapid is meaningful because by its very nature a research project can help accelerate mobilization for the full range of mobilization actors, including other research projects. As such, it is not surprising that they can become temporal conduits to knowledge mobilization in all of its forms.

6 Conclusions

In this paper, we proposed the use of a temporal betweenness measure (foremost betweenness) to analyse a knowledge mobilization network that had been already studied using

Table 5: Major invisible brooks

Actor	Time	Stat. Rank	Temp. Rank	Type
J3(08)	[08-11]	9	117	journal
	[09-11]	12	84	
C3(11)	[08-11]	10	191	citing publication
	[09-11]	15	153	
C4(11)	[08-11]	15	105	citing publication
H2(05)	[06-11]	16	118	researcher
	[07-11]	15	134	
A3(07)	[08-11]	16	187	publication
C5(07)	[08-11]	18	158	citing publication

classical "static" parameters. Our goal was to see the impact on the perceived static central nodes when employing a measure that explicitly takes time into account. We observed interesting differences. In particular, we witnessed the emergence of invisible rapids: nodes whose static centrality was considered negligible, but whose temporal centrality appears relevant. Our interpretation is that nodes with high temporal betweenness contribute to accelerate mobilization flow in the network and, as such, they can remain undetected when the analysis is performed statically. We conclude that foremost betweenness is a crucial tool to understand the temporal role of the actors in a dynamic network, and that the combination of static and temporal betweenness is complementary to provide insights into their importance and centrality.

Temporal network analysis as performed here is especially pertinent for KM research that must take time into account to understand academic research impact beyond the narrow short-term context of academia. Measures of temporal betweenness, as studied in this paper, can provide researchers and funders with critical tools to more confidently investigate the role of specific mobilization actors for short and long-term impact within and beyond academia. The same type of analysis could clearly be beneficial when applied to any other temporal context.

In conclusion, we focused here on a form of temporal betweenness designed to detect accelerators. This is only a first step towards understanding temporal dimensions of social networks; other measures are already under investigation.

Acknowledgment

This work was partially supported by NSERC Discovery Grant and by Dr. Flocchini's University Research Chair.

References

[1] F. Amblard, A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. On the temporal analysis of scientific network evolution. *International Conference on Computational Aspects of Social Networks* (CASoN), 169-174, 2011.

[2] S. Beaudoin, X. Roucou X. Genetic prion mutants inhibit two RNA granules, P-Bodies and stress granules. *PrPCANADA*, Ottawa, Canada, 2010.

[3] C. Binz, B. Truffer, and L. Coenen. Why space matters in technological innovation systems Mapping global knowledge dynamics of membrane bioreactor technology. *Research Policy*, 43(1):138–155, February 2014.

[4] B Bui Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

[5] W. P. Boland, P. W.B. Phillips, C. D. Ryan, and S. McPhee-Knowles. Collaboration and the Generation of New Knowledge in Networked Innovation Systems: A Bibliometric Analysis. *Procedia - Social and Behavioral Sciences*, 52:15–24, 2012.

[6] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility. *Proc. 6th IFIP conference on Theoretical Computer Science*, 111–124, 2010.

[7] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring Temporal Lags in Delay-Tolerant Networks. *IEEE Transactions on Computers*, 63(2), 397–410, 2014.

[8] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[9] K. Chan and J. Liebowitz. The synergy of social network analysis and knowledge mapping: a case study. *International Journal of Management and Decision Making*, 7(1):19–35, 2006.

[10] M.J. Eppler. Making knowledge visible through intranet knowledge maps: concepts, elements, cases. *Proc. 34th Annual Hawaii International Conference on System Sciences*, I 2001.

[11] L. C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 35–41, 1977.

[12] A. Galati, V. Vukadinovic, M. Olivares, and S. Mangold. Analyzing temporal metrics of public transportation for designing scalable delay-tolerant networks, *Proc. 8th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, 37-44, 2013

[13] J. Gaudet. It takes two to tango: knowledge mobilization and ignorance mobilization in science research and innovation. *Prometheus*, 13(3):169-187, 2013.

[14] J. Gaudet. The Mobilization-Network Approach for the Social Network Analysis of Knowledge Mobilization in Science Research and Innovation. *uO Research*, (PrePrint):1–28, 2014.

[15] S. Gómez, P. Jensen, and A. Arenas. Analysis of community structure in networks of correlated data. *Physical Review E*, 80(1), 2009.

[16] E.P.C. Jones, L. Li, J.K. Schmidtke, and P.A.S. Ward. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 6(8), 943–959, 2007.

[17] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network, *Proc. 14th International Conference on Knowledge Discovery and Data Mining* (KDD), 435–443, 2008.

[18] V. Kostakos. Temporal graphs. *Physica A*, 388(6), 1007–1023, 2009.

[19] H. Kim and R. Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.

[20] N. L Klenk, A. Dabros, and G. M Hickey. Quantifying the research impact of the Sustainable Forest Management Network in the social sciences: a bibliometric study. *Canadian Journal of Forest Research*, 40(11):2248–2255, 2010.

[21] M. Konschake, H. HK Lentz, F. J Conraths, P. Hövel, and Thomas Selhorst. On the robustness of in-and out-components in a temporal network. *PloS one*, 8(2), 2013.

[22] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

[23] F. Kuhn, N.Lynch and R. Oshman. Distributed computation in dynamic networks, *Proc. 42nd ACM Symposium on Theory of Computing* (STOC), 513–522, 2010.

[24] H. H. K. Lentz, T. Selhorst, and I. M. Sokolov. Unfolding accessibility provides a macroscopic approach to temporal networks. *Phys. Rev. Lett.*, 110:118701–118706, 2013.

[25] C. Liu, and J. Wu. Scalable Routing in Cyclic Mobile Networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(9),1325–1338, 2009.

[26] O. Michail, I. Chatzigiannakis, and P. Spirakis. Distributed computation in dynamic networks, *Journal of Parallel and Distributed Computing*, 74(1), 2016–2026, 2014.

[27] A. Y. Mutlu, E. Bernat, and S. Aviyente. A signal-processing-based approach to time-varying graph analysis for dynamic brain network identification. *Computational and mathematical methods in medicine*, 2012.

[28] W. Quattrociocchi, R. Conte, and E. Lodi. Opinions manipulation: Media, power and gossip. *Advances in Complex Systems*, 14(4):567–586, 2011.

[29] H. Saba, V. C Vale, M. A Moret, and J-G Miranda. Spatio-temporal correlation networks of dengue in the state of bahia. *BMC Public Health*, 14(1):1085, 2014.

[30] N. Santoro, W. Quattrociocchi, P. Flocchini, A. Casteigts, and Frédéric Amblard. Time-Varying Graphs and Social Network Analysis: Temporal Indicators and Metrics. *Proc. of 3rd AISB Social Networks and Multiagent Systems Symposium (SNAMAS)*, 32–38, 2011.

[31] J. Saramaki P. Holme. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[32] J. Tang, M. Musolesi, C. Mascolo, and V. Latora. Temporal distance metrics for social network analysis. *Proc. 2nd ACM Workshop on Online Social Networks* (WOSN), 31-36, 2009.

[33] C. Tantipathananandh, T. Berger-Wolf, D. Kempe. A framework for community identification in dynamic social networks, *Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 717-726, 2007

[34] Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3) pp 410-421.