J Math Model Algor (2011) 10:371–391
DOI 10.1007/s10852-011-9163-3

Snapshot Centrality Indices in Dynamic FIFO Networks

Tatiana M. Tabirca · Kenneth N. Brown · Cormac J. Sreenan

Received: 28 September 2010 / Accepted: 1 September 2011 / Published online: 20 September 2011
© Springer Science+Business Media B.V. 2011

Abstract The article introduces the concept of snapshot dynamic indices as centrality measures to analyse how the importance of nodes changes over time in dynamic networks. In particular, the dynamic stress-snapshot and dynamic betweenness snapshot are investigated. We present theoretical results on dynamic shortest paths in first-in first-out dynamic networks, and then introduce some algorithms for computing these indices in the discrete-time case. Finally, we present some experimental results exploring the algorithms' efficiency and illustrating the variation of the dynamic betweenness snapshot index for some sample dynamic networks.

Keywords Dynamic networks · Centrality indices · Network applications

1 Introduction

Centrality indices were first defined over 50 years ago to represent the importance of a node in a graph, initially based on counting the number of shortest paths that use the node [21]. The concept has been developed extensively since then, with new definitions, practical applications, theoretical results and efficient computation

T. M. Tabirca (✉) · C. J. Sreenan
Mobile and Internet System Laboratory, Department of Computer Science,
University of College Cork, Cork, Ireland
e-mail: tabirca1@cs.ucc.ie

C. J. Sreenan
e-mail: cjs@cs.ucc.ie

K. N. Brown
Cork Constraint Computation Centre, Department of Computer Science,
University of College Cork, Cork, Ireland
e-mail: k.brown@cs.ucc.ie

algorithms. Betweenness centrality [12] was introduced for the analysis of social networks, and used a weighting factor based on the number of different shortest paths between all pairs of nodes. If we assume any communication between two nodes flows along the shortest path, then betweenness estimates the number of those communications that visit a particular node, and thus it estimates the importance of that node to the network. This was naturally extended to the concept of arc-betweenness centrality and used to generate communities within a network [18]. Brandes developed an efficient algorithm for computing betweenness centrality and showed this could be extended to other variations of betweenness [5].

All of these centrality indices assumed that the graph is static, and that the shortest paths do not change. However, many practical problems require dynamic graphs, in which the node and arc sets or the arc costs vary over time. For example, road traffic networks can be modeled as a graph in which nodes are intersections and arcs represent roads. The average time taken to traverse an arc depends on the amount of traffic on the road, which varies throughout the day. As congestions increase over time, a superficially longer route may become the shortest path between two points, and thus the centrality of some nodes can change. Similar patterns appear in other practical examples, including electricity distribution networks, telecommunications networks, and for navigation networks where the arc costs vary independently of traffic, such as evacuation from a fire, or where weather patterns affect travel time.

In many of these cases, it is important to know how the centrality of a node varies over time, for example in route planning or for dynamic repair of a communication networks. The centrality measures should therefore be adapted to include a dynamic element which represents time. For indices which are based on shortest paths, there are two main time elements to consider. Firstly, we may wish to know the centrality of nodes in terms of the shortest paths originating at some time t. More importantly, for network analysis based on congestion or workload, we may wish to know which node will be visited at time t by the greatest number of shortest paths, regardless of when they orginated. This measure can be considered to be a snapshot of the network, showing importance of a node at a particular time. The research problem is how to define appropriate *snapshot centrality* measures, and how to compute them efficiently.

We introduce two dynamic snapshot centrality indices: *stress and betweenness*, which are both based on dynamic shortest paths. Certainly, our approach can be used to some other types of importance indices e.g. graph [15] or closeness [20]. Firstly, we present some theoretical results for first-in first-out (FIFO) dynamic networks, which are used to connect the dynamic snapshot indices with the number of dynamic shortest paths between nodes. We then provide an original scheme for discretising a continuous dynamic network problem, followed by an algorithm for computing the indices for discrete-time problems. The algorithm firstly uses an adapted computation of the dynamic shortest paths, which also generates the number of the shortest paths for the all-to-all case. Our algorithm is exact for discrete problems, and approximate for continuous problems. Finally, we present some experimental evaluations of the algorithm. We show how the algorithms performance varies with the size of network, and with the granularity of the discretisation. Finally, we show how the dynamic snapshot betweenness changes over time for three different network types, including both cyclic traffic and evacuation problems.

2 Literature Review

This section briefly reviews the main research on centrality indices. We assume an undirected graph $G = (V, A)$ for which there is a cost function $c : A \to [0, \infty)$. For any two nodes $u, v \in V$, we define the following elements:

$$d(u, v) = \text{the cost of the shortest path between } u \text{ and } v. \tag{1}$$

$$\sigma(u, v) = \text{the number of shortest paths between } u \text{ and } v. \tag{2}$$

$$\sigma(u, w; v) = \text{the number of shortest paths between } u \text{ and } w \text{ that pass } v. \tag{3}$$

By convention we can consider $d(u, u) = 0$ and $\sigma(u, u) = 1$.

2.1 Static Centrality Indices

Definition 1 The static stress and betweenness centrality indices can be defined for the graph $G = (V, A)$ as follows:

$$C_S(v) = \sum_{u \neq v \neq w} \sigma(u, w; v) \text{ stress centrality [21]} \tag{4}$$

$$C_B(v) = \sum_{u \neq v \neq w} \frac{\sigma(u, w; v)}{\sigma(u, w)} \text{ betweenness centrality [12].} \tag{5}$$

When the denominator $\sigma(u, w)$ from Eq. 5 is 0 then $\sigma(u, w; v)$ is 0 too and by convention we consider $\frac{\sigma(u,w;v)}{\sigma(u,w)} = 0$.

The static betweenness centrality has been used mostly in the study of social networks [12], while the stress centrality has had multiple application to transmission networks. However, both measures show the degree of connectivity between v and all the other nodes when shortest paths are used. The stress centrality index counts the number of shortest paths that go through a node. If we consider for example a communication network then this index can identify the work or the stress a node has to take in communication. Finally, for networks where there may be multiple shortest paths of the same length, the betweenness centrality index computes the percentage of shortest paths on which the node lies.

The computation of these values is closely related to the computation of shortest paths in the All-to-All case. Any All-to-All algorithm for shortest paths could be applied to generate the values $d(u, v)$; however Dijkstra's algorithm offers the most efficient solution with the complexity $O\left(m \cdot n + n^2 \cdot \log n\right)$ [2]. Jacob et al. [16] reviewed the main methods to calculate the centrality indices and concluded that the most efficient solution is a generic computation with three steps. The first step is to calculate all the values $d(u, v), \forall u, v \in V$ based on the All-to-All Dijkstra's algorithm. Then the second step computes the values $\sigma(u, w)$ and $\sigma(u, w; v)$ based on a simple graph traversal computation. For example, for each pair of distinct nodes $u, w \in V$, we can consider all the neighbours v of u so that $(u, v) \in A$; when $d(u, w) = c(u, v) + d(v, w)$ we have a shortest path between u and w that goes through v so that the value $\sigma(u, w)$ is increased with $\sigma(v, w)$. This second step has

therefore a complexity of $O\left(m \cdot n + n^2\right)$ because of the nature of graph traversal. The third step is to compute directly the centrality indices from Eqs. 4 and 5 by iterating through one or two indices which would increase the complexity to $O\left(n^3\right)$. Therefore, this generic computation in three steps generates the centrality indices from Eqs. 4 and 5 in a total complexity of $O\left(n^3\right)$.

However, Brandes [5] provided an efficient way to compute the betweenness based on the notion of pair dependency $\delta(s, t; v) = \frac{\sigma(s,t;v)}{\sigma(s,t)}$. Brandes proved that $\delta(s; v) = \sum_{s \neq v \neq t} \delta(s, t; v)$, the sum of all pair dependencies from the source s, satisfies

$$\delta(s; v) = \sum_{w:\ v \in P_s(w)} \frac{\sigma(s, v)}{\sigma(s, w)} \cdot (1 + \delta(s; w)), \tag{6}$$

where $P_s(w) = \{v : (v, w) \in A, d(s, w) = d(s, v) + c(v, w)\}$. Based on Eq. 6, Brandes developed an efficient computation in $O\left(m \cdot n + n^2 \cdot \log n\right)$ to generate the betweenness centrality index [5]. Moreover, Brandes extended this approach to various other types of betweenness centrality to include situations like arc betweenness, group betweenness, proxy betweenness [6]. Unfortunately, the Brandes approach cannot be applied to some other types of centrality indices as it only deals with the betweenness centrality.

2.2 Dynamic Shortest Paths and Dynamic Indices

All these above indices are defined for static graphs. However, concrete practical problems are modeled with complex networks, which are in many cases dynamic. Therefore, the concept of dynamic centrality indices is needed to investigate these problems. Usually, a dynamic graph or network is given by a sequence of graphs over time

$$G_t = (V_t, A_t),\ t = 0, 1, ..., T$$

or by one graph $G = (V, A)$ for which the costs of the arcs A

$$c^{(t)} : A \to [0, \infty),\ t = 0, 1, ..., T$$

are variable in time. Similarly to the static case, the problem of dynamic centrality indices is based on dynamic shortest paths. This dynamic shortest paths problem is a classical topic in combinatorial optimisation investigated since late 60's e.g Cooke et al. [9]. Dynamic shortest paths more recently became very popular with the emergence of Intelligent or Dynamic Transportation Systems e.g. Chabini [7, 8], Ahuja [3] or Demetrescu [11]. There are two well established approaches to this problem, which are quite different in essence. The first approach considers some dynamic equations for the numbers $d^{(t)}(u, v)$, the cost of the path from u to v starting at time t, to reflect the evolution in time of the shortest paths costs. These equations are then processed using dynamic programming techniques mainly in a retrospective way e.g. the values $d^{(t)}(u, v)$, $d^{(t+1)}(u, v)$, ... are used to generate the value $d^{(t-1)}(u, v)$ [7]. This approach is useful when we want to consider all the cost changes that take place at time t. In this way we keep the time variable t in a central place in the computation. This method is very effective when the time interval to investigate

the dynamic network is relatively small compared to the network size. The second approach is to preserve the shortest paths when changes take place in the graphs e.g. removing a node or an arc or changing an arc's cost. This approach was introduced by Ramalingam et al. [19] and then refined successively by various contributors; see [11] for a complete review. The emphasis of this approach is to update efficiently the shortest path spanning tree when a change takes place. This method does not consider the time variable t in the dynamicity of the changes and it is very effective when the number of changes is small.

There is little published research on dynamic centrality indices despite the large body of work on the dynamic shortest path problem. However, some work has been done on the applications of these indices to various types of concrete problems. Gross et al. [13] presented a good review of practical problems from engineering and transportation networks to social networks. A more specialised study is presented by Abraham et al. [1] to cover a variety of topics associated with dynamic social networks. A commonality of these works is that they investigate these problems rather statically where some static centrality measures are applied to study the network at a moment in time. This means that the centrality measure does not consider any information coming from past states of the network. This approach is perhaps generated by the lack of results, including computational tools to calculate the dynamic centrality indices.

Habiba et al. [14] extended the concept of betweenness centrality to dynamic networks. They consider a dynamic network as a series of graphs $G_t = (V_t, A_t),\ t = 0, 1, ..., T$ that change nodes and arcs over time. In this dynamic network the arc costs are all 1. A temporal path in this dynamic network is a sequence of nodes $P = (u_0, u_1, ..., u_p)$ so that each arc (u_{i-1}, u_i) is available in the network before the arc (u_i, u_{i+1}). This definition allows the situation when delays are possible in nodes e.g. in the node u_i we can wait for the arc (u_i, u_{i+1}) to be available. Based on this definition the authors derived some equations for the dynamic betweenness centrality which are similar to Eq. 6. However, the article does not present a clearly-stated algorithm or scheme to compute the dynamic equations. Secondly, the authors transform the dynamic graph into a time-expanded static graph with $n' = T \cdot n$ nodes and $m' = 2 \cdot T \cdot m + T \cdot n$ arcs. This static graph is then traversed for each node v to generate the set of nodes used in the dynamic equation (Eq. 6). This traversal however introduced a complexity of $O\left(n'^2 + n' \cdot m'\right) = O\left(T^2 \cdot n^2 + T^2 \cdot n \cdot m\right)$, which is quadratic in T. Thirdly and perhaps most importantly, this dynamic betweenness does not reflect how a node is used in the shortest paths at the current time t but how the node will be used in the future on shortest paths starting at t. Furthermore, the authors did not offer any evaluation of the complexity nor about the execution times obtained in the examples they presented. Finally, the work covers only the situation when the arcs have the cost 1, and therefore it is difficult to apply this approach to other types of dynamic networks whose costs vary over time.

A different approach to centrality betweenness was introduced by Lerman et al. [17] on dynamic networks. The authors associated probabilities to some types of events that can occur in dynamic networks e.g. the probability that one node initiates a message or the probability that one node forwards a received message to its neighbors at a given time or probability that one node stores a message for a given time. They then introduced two new types of centrality indices called *cumulative dynamic centrality* and *retained cumulative dynamic centrality* to reflect

the probability that a message sent by a node in a network reaches another after some period of time. Algorithms were also introduced to generate these indices over a sequence of time, and the authors then illustrated how these two dynamic indices work on a citation network. This approach is not centered on shortest path connections but rather on probabilistic connections, and so the time the messages travel between nodes is not encapsulated in the computation.

3 Dynamic Shortest Path Computation

The idea of dynamic centrality indices depends on dynamic shortest paths in dynamic networks, and practical computation of the indices will require efficient algorithms for finding dynamic shortest paths. Therefore we now consider some theoretical aspects of dynamic shortest paths, and concentrate on the FIFO case. We consider the issues of continuous-time problems versus discrete-time problems, and we propose a scheme for approximating a continuous-time problem by a discrete-time one. Finally, we then present our algorithms for computing the dynamic shortest paths.

We represent a dynamic network as $G^{(t)} = \left(G = (V, A), c^{(t)}\right)$, where $G = (V, A)$ is the underlying static graph and $c^{(t)} : A \to [0, \infty) \cup \{\infty\}$ is a dynamic cost function. The dynamic network is investigated over the continuous time $t \in [0, \infty)$. The value $c^{(t)}(u, v)$ represents the cost of the arc (u, v) at the time t, which can be interpreted as the time taken to traverse the arc at the time t. We use $c^{(t)}(u, v) = \infty$ when the arc (u, v) is no longer available in the dynamic network.

Based on the dynamic network $G^{(t)} = \left(G = (V, A), c^{(t)}\right)$, we can consider the dynamic shortest paths between two nodes. The cost of the path $P = \left(u_0, u_1, ..., u_p\right)$ starting from u_0 at time t to u_p is

$$Cost^{(t)}(P) = c^{(t)}(u_0, u_1) + c^{(t_1)}(u_1, u_2) + ... + c^{(t_{p-1})}(u_{p-1}, u_p), \tag{7}$$

where the times satisfy

$$t_1 = t + c^{(t)}(u_0, u_1), ..., t_{p-1} = t_{p-2} + c^{(t_{p-2})}(u_{p-2}, u_{p-1}). \tag{8}$$

Equation 7 illustrates that if a path or transmission leaves u_0 at time t to reach u_1 at time $t_1 = t + c^{(t)}(u_0, u_1)$ then it can leave u_1 at the time t_1 for u_2 and so on. We can simplify this equation if we consider the arrival time function $f_a : [0, \infty) \to [0, \infty)$, $f_a(t) = t + c^{(t)}(a)$ for each arc a. If $a = (u, v)$ is an arc between u, v then the value $f_a(t)$ represents the arrival time in v if we depart u at the time t. If we consider that the path P as a sequence of arcs $P = \left(a_0, a_1, ..., a_{p-1}\right)$ then the cost of P can be re-written as a composition of arrival time functions as follows $Cost^{(t)}(P) = \left(f_{a_{p-1}} \circ ... \circ f_{a_0}\right)(t) - t$ [10].

The dynamic shortest path between u_0 and u_p at time t is the path P that minimizes the cost $Cost^{(t)}(P)$ of Eq. 7, hence we can speak about the dynamic shortest path cost given by

$$d^{(t)}(u_0, u_p) = \min\left\{Cost^{(t)}(P) : \ P = (u_0, ..., u_p) \text{ path between } u_0 \text{ and } u_p\right\} \tag{9}$$

We can also define $\sigma^{(t)}(u_0, u_p)$ as the number of shortest paths between u_0 and u_p, that leave u_0 at the time t. Similarly, $\sigma^{(t)}(u_0, u_p; v)$ is the number of shortest paths

that between u_0 and u_p departing u_0 at the time t that go through the node v. One can see that these shortest paths leave u_0 at the time t and they will reach the node v in the future. However, many practical applications need to know how the node v is used in shortest paths *now* at the current time t and not in the future. For that we introduce $\sigma^{(t')}(u_0, u_p; v) \downarrow t$ as notation for the number of shortest paths between u_0 and u_p that leave u_0 at time t' to go to u_p and pass through v at the time t.

3.1 Dynamic Shortest Paths in the FIFO Case

We now consider the particular case when the dynamic costs $c^{(t)}(u, v)$ satisfy the FIFO rule

$$t < t' \Rightarrow t + c^{(t)}(u, v) < t' + c^{(t')}(u, v), \ \forall (u, v) \in A. \tag{10}$$

This is also called the non-overtaking rule and it ensures that commodities travel along the arc (u, v) in a First-In First-Out manner. The FIFO rule applies to many dynamic networks that we encounter in real life problems e.g. transportation networks, information transmission networks etc. Note that the arrival time functions f_a are increasing when the FIFO rule holds. In the following results, we consider the continuous case where the time $t \in [0, \infty)$ is real.

Lemma 1 *In the FIFO case, if $P = (u_0, ..., u_k, ..., u_p)$ is a shortest path between u_0 to u_p starting from u_0 at the time t_0 then*

1. *$P_1 = (u_0, ..., u_k)$ is a shortest path starting from u_0 at the time t_0*
2. *$P_2 = (u_k, ..., u_p)$ is a shortest path too starting from u_k at the time $t_0 + d^{(t_0)}(u_0, u_k)$, respectively.*

Proof Suppose that $P_1 = (u_0, u_1, ..., u_k)$ is not optimal so that there is another path $P'_1 = (u_0 = u'_0, u'_1, ..., u'_k = u_k)$ so that $Cost^{(t_0)}(P'_1) < Cost^{(t_0)}(P_1)$. If the path P_1 leaves u_0 at the time t_0 then u_k is reached at the time $t_k = t_0 + Cost^{(t_0)}(P_1)$. Similarly, u_k is reached at the time $t'_k = t_0 + Cost^{(t_0)}(P'_1)$ if the path P'_1 is used. Since $t'_k < t_k$ we can iteratively apply the FIFO rule as follows:

$$t'_k < t_k \Rightarrow t'_{k+1} = t'_k + c^{(t'_k)}(u_k, u_{k+1}) < t_{k+1} = t_k + c^{(t_k)}(u_k, u_{k+1})$$

$$t'_{k+1} < t_{k+1} \Rightarrow t'_{k+2} = t'_{k+1} + c^{(t'_{k+1})}(u_{k+1}, u_{k+2}) < t_{k+2} = t_{k+1} + c^{(t_{k+1})}(u_{k+1}, u_{k+2})$$

$$...$$

$$t'_{p-1} < t_{p-1} \Rightarrow t'_p = t'_{p-1} + c^{(t'_{p-1})}(u_{p-1}, u_p) < t_p = t_{p-1} + c^{(t_{p-1})}(u_{p-1}, u_p).$$

If we add up all these equations, we find that

$$t'_k + Cost^{(t'_k)}(P_2) < t_k + Cost^{(t_k)}(P_2) \Rightarrow$$

$$t_0 + Cost^{(t_0)}(P'_1) + Cost^{(t'_k)}(P_2) < t_0 + Cost^{(t_0)}(P_1) + Cost^{(t_k)}(P_2) \Rightarrow$$

$$t_0 + Cost^{(t_0)}(P'_1, P_2) < t_0 + Cost^{(t_0)}(P_1, P_2),$$

therefore the cost of the path (P'_1, P_2) is less than the cost of the optimal path.

The second part of the proof is straightforward based on the fact the path P_1 is optimal and it reaches the node u_k at the time $t_k = t_0 + d^{(t_0)}(u_0, u_k)$. Suppose that P_2 is not optimal so that there is a dynamic path P_2' so that $Cost^{(t_k)}(P_2') < Cost^{(t_k)}(P_2)$. In this case the path $P' = (P_1, P_2')$ satisfies

$$t_0 + Cost^{(t_0)}(P_1) + Cost^{(t_k)}(P_2') < t_0 + Cost^{(t_0)}(P_1) + Cost^{(t_k)}(P_2),$$

which means that it is shorter than the optimal path (P_1, P_2). □

Lemma 2 *If the values $\sigma^{(t)}(u, v)$ are known then $\sigma^{(t)}(u, w; v)$ satisfies*

$$\sigma^{(t)}(u, w; v) = \epsilon^{(t)}(u, w; v) \cdot \sigma^{(t)}(u, v) \cdot \sigma^{(t+d^{t}(u,v))}(v, w), \tag{11}$$

where $\epsilon^{(t)}(u, w; v) = \begin{cases} 1 & \text{if } d^{(t)}(u, w) = d^{(t)}(u, v) + d^{(t+d^{(t)}(u,v))}(v, w) \\ 0 & \text{otherwise} \end{cases}$.

Proof Note that all the dynamic shortest paths leave u at the time t and they all arrive to v at the time $t' = t + d^{(t)}(u, v)$. Two situations are considered in the following. The first case is when the node v does not belong to any dynamic shortest path between u and w, which happens when $d^{(t)}(u, w) \neq d^{(t)}(u, v) + d^{(t')}(v, w)$. In this case, we have $\epsilon^{(t)}(u, w; v) = 0$ and $\sigma^{(t)}(u, w; v) = 0$ so the Eq. 11 holds.

The second case reflects the situation when the node v satisfies $d^{(t)}(u, w) = d^{(t)}(u, v) + d^{(t')}(v, w)$ e.g. it belongs to some dynamic shortest paths between u and w. Consider a dynamic shortest path $P = (u, a, ..., v, b, .., w)$ from the node u to the node w at the time t that passes through v. According to Lemma 1, this path P generates a dynamic shortest path $P_1 = (u, a, ..., v)$ between u and v at the time t and also a dynamic shortest path $P_2 = (v, b, .., w)$ between v and w at the time t'. Similar arguments can be made to show that from a shortest path P_1 between u and v and a shortest path P_2 between v and w we can construct a shortest path $P = (P_1, P_2)$ between u and w that passes the node v. If we count all such shortest paths we find that $\sigma^{(t)}(u, w; v) = \sigma^{(t)}(u, v) \cdot \sigma^{(t')}(v, w)$, therefore Eq. 11 holds. □

Lemma 3 *In the FIFO case, the function $t \mapsto t + d^{(t)}(u, v)$ is increasing for any arc $(u, v) \in A$.*

Proof Consider a path $P = (a_0, a_1, ..., a_{p-1})$ as a sequence of arcs, whose cost satisfies $t + Cost^{(t)}(P) = \left(f_{a_{p-1}} \circ ... \circ f_{a_0}\right)(t)$. Since the arrival time functions are all increasing in the FIFO case, we can find that $t < t' \Rightarrow t + Cost^{(t)}(P) < t' + Cost^{(t')}(P)$. If P is a dynamic shortest path between u and v at the time t' we have

$$\begin{aligned} t + Cost^{(t)}(P) &< t' + Cost^{(t')}(P) = t' + d^{(t')}(u, v) \Rightarrow \\ t + d^{(t)}(u, v) &\leq t + Cost^{(t)}(P) < t' + d^{(t')}(u, v) \Rightarrow \\ t + d^{(t)}(u, v) &< t' + d^{(t')}(u, v), \end{aligned}$$

which means that the function $t \mapsto t + d^{(t)}(u, v)$ is increasing. □

Denote with $T^{(t)}(u, v) = \{t' : t = t' + d^{(t')}(u, v)\}$ the set of times from the past from which we can reach v at the time t coming from u. Then Lemma 3 establishes that the set $T^{(t)}(u, v)$ is either empty or has only one element. The combination between Lemmas 2 and 3 also provides that

$$\sigma^{(t')}(u, w; v) \downarrow t = \begin{cases} \sigma^{(t')}(u, v) \cdot \sigma^{(t)}(v, w) & \text{if } t' \in T^{(t)}(u, v), \\ 0 & \text{if } t' \notin T^{(t)}(u, v), \end{cases} . \tag{12}$$

Theorem 1 Chabini [7] *In the FIFO case, the values $d^{(t)}(u, v)$ satisfy the equation*

$$d^{(t)}(u, v) = \begin{cases} \min\{c^{(t)}(u, w) + d^{(t+c^{(t)}(u,w))}(w, v) : (u, w) \in A\} & \text{if } u \neq v \\ 0 & \text{if } u = v \end{cases} \tag{13}$$

The results presented above only hold in the FIFO case. For non-FIFO costs, it can be possible that a sub-path of a dynamic shortest path is not optimal. This means that Eq. 11 cannot hold true for non-FIFO costs so that we do not have a simple way to calculate the dynamic values $d^{(t)}(u, v)$, $\sigma^{(t)}(u, v)$, $\sigma^{(t)}(u, w; v)$ and $\sigma^{(t')}(u, w; v) \downarrow t$. More importantly, Dean [10] showed that the computation of the $d^{(t)}(u, v)$ values in the non-FIFO case is NP-hard as it can be reduced to some variations of the knapsack problem. However, if we relax Eq. 8 to allow delays in nodes then we can compute these values in polynomial time [7].

3.2 Discrete-time Computation

The process of making the continuous time interval discrete is important for having efficient algorithms. Previous work on dynamic shortest paths depended on continuous cost functions, and on the time interval being $[0, \infty)$ generating algorithms that are not efficient. In that research, the dynamic costs are considered as integers in order to have the continuous case equations with $t \in [0, \infty)$ translated into similar equations in the discrete case with the time $t \in \{0, 1, 2, ...\}$ [8, 10]. In this section we introduce an accurate method to transform the equations in the continuous space into equations in the discrete space. In particular, for computing the snapshot indices, it is important to identify exactly when a path arrives at an intermediate node. Therefore, we present method for discretizing a continuous time problem, which will allow us to define efficient algorithms for computing the dynamic shortest paths and indices.

The main input for our investigation is given by a dynamic network $G^{(t)} = \left(G = (V, A), c^{(t)}\right)$ with the dynamic costs $c^{(t)}$, $t \in [0, t_{\max}]$, which are under the FIFO rule. The costs $c^{(t)}$ are considered static for all the times $t \geq t_{\max}$ and they are given by $c^{(t)} = c^{(t_{\max})}, t \geq t_{\max}$. Sometimes, the dynamic network can be investigated in its evolution over the interval $[0, t_{\max}]$. However, most of the time this approach is not possible especially in practical applications where the dynamic network is analysed during the times $t \in \{0, 1, ..., t_{\max}\}$, for which the dynamic costs $c^{(t)}(a)$, $t \in \{0, 1, ..., t_{\max}\}$ are given by some observations, or by some readings, or even by some simulations. In this case the input is given by the dynamic network $G^{(t)} = \left(G = (V, A), c^{(t)}\right)$ with the dynamic costs $c^{(t)}$, $t \in \{0, 1, ..., t_{\max}\}$, which satisfy the FIFO rule. The plan is to extend the discrete FIFO costs $c^{(t)}$ to the cost function $C^{(t)}, t \geq 0$ which is FIFO too and satisfies $C^{(t)}(a) = c^{(t)}(a), t \in \{0, 1, ..., t_{\max}\}$. This extension can be done in various ways but, for simplicity, we consider $C^{(t)}(a)$ as a

sequence of line segments between two consecutive points of the set $\{(t, c^{(t)}(a))\, t \in \{0, 1, ..., t_{\max}\}\}$. This means that the cost function $C^{(t)}(a)$ is defined by the equation

$$C^{(t)}(a) = \left(c^{(k+1)}(a) - c^{(k)}(a)\right) \cdot (t - k) + c^{(k)}(a), \text{ if } t \in [k, k+1], \tag{14}$$

when $k \in \{0, 1, ..., t_{\max} - 1\}$ and $a \in A$. It is clear that the function $C^{(t)}(a)$ satisfies $C^{(t)}(a) = c^{(t)}(a),\ t \in \{0, 1, ..., t_{\max}\}$.

Theorem 2 *The function $C^{(t)}(a)$ defined by Eq. 14 satisfies the FIFO rule.*

Proof We consider two cases in order to prove that this function is still FIFO. Firstly, when $k \le t < t' \le k + 1$, we have the following equivalences

$$\begin{aligned} t + C^{(t)}(a) &< t' + C^{(t')}(a) \iff \\ t + \left(c^{(k+1)}(a) - c^{(k)}(a)\right) \cdot (t - k) + c^{(k)}(a) \\ < t' + \left(c^{(k+1)}(a) - c^{(k)}(a)\right) \cdot (t' - k) + c^{(k)}(a) &\iff \\ t' - t + \left(c^{(k+1)}(a) - c^{(k)}(a)\right) \cdot (t' - t) > 0 &\iff \\ \left(c^{(k+1)}(a) - c^{(k)}(a) + 1\right) \cdot (t' - t) > 0, \end{aligned}$$

which is true because $c^{(k+1)}(a) - c^{(k)}(a) + 1 > 0$ from the FIFO rule. Secondly, $k_1, k_2 \in \{0, 1, ..., t_{\max}\}$, are two indices so that $k_1 = \lceil t \rceil$ and $k_2 = \lfloor t' \rfloor$, which gives that $t \le k_1 \le k_2 \le t'$. Note that at least one inequality must be strict. In this case we can apply the first case as follows

$$\begin{aligned} t \le k_1 &\Rightarrow t + C^{(t)}(a) \le k_1 + C^{(k_1)}(a) \\ k_2 \le t' &\Rightarrow k_2 + C^{(k_2)}(a) \le t' + C^{(t')}(a) \end{aligned}$$

and the FIFO rule for the discrete costs

$$k_1 \le k_2 \Rightarrow k_1 + C^{(k_1)}(a) = k_1 + c^{(k_1)}(a) \le k_2 + c^{(k_2)}(a) = k_2 + C^{(k_2)}(a),$$

which produces $t + C^{(t)}(a) \le k_1 + C^{(k_1)}(a) \le k_2 + C^{(k_2)}(a) \le t' + C^{(t')}(a)$. Note that at least one inequality is strict so that the function $C^{(t)}(a)$ satisfies the FIFO rule for this second case. Therefore, the cost function $C^{(t)}(a)$ is under the FIFO rule so that we can apply Eq. 13 to generate the function $d^{(t)}(a)$. □

Hence, we can assume that the input for our problem is given by a dynamic network in which the dynamic costs are under the FIFO rule over the continuous time interval $t \ge 0$. In the following we will use $C^{(t)}(a)$ as notation for the dynamic costs.

A simple way to discretize the time $t \ge 0$ is to work with the discrete interval $t \in \{0, 1, ..., t_{\max}\}$. This approach is feasible when the dynamic costs $C^{(t)}(a)$ are all integers so that the upper index $t + C^{(t)}(a)$ from Eq. 13 is an integer too. However, the discrete interval $t \in \{0, 1, ..., t_{\max}\}$ does not suffice when some of the costs $C^{(t)}(a)$ are real numbers and a finer process of discretization is needed to generate a discrete set of times T so that $t + C^{(t)}(a) \in T$ when $t \in T$. For that we assume that the values of the function $t \mapsto C^{(t)}(a)$ are approximated with a given error ϵ so that $\frac{1}{\epsilon} \in N$. In this case the costs $C^{(t)}(a)$ are all rational numbers so that $C^{(t)}(a) \cdot \frac{1}{\epsilon} \in N$.

The discrete time interval to model the continuous time is now provided by $T = \{0, \epsilon, 2 \cdot \epsilon, ..., \epsilon \cdot l = t_{\max}\}$, where $l = t_{\max} \cdot \frac{1}{\epsilon}$. We can now re-write Eq. 13 using only times from T to get

$$d^{(k\cdot\epsilon)}(u, v) = \min\{C^{((k\cdot\epsilon)}(u, w) + d^{(k\cdot\epsilon+C^{(k\cdot\epsilon)}(u,w))}(w, v) : (u, w) \in A\}, \tag{15}$$

when $u \neq v$. Moreover, if we denote $d^k(u, v) = d^{(k\cdot\epsilon)}(u, v)$ then Eq. 15 can generate the following equation over the discrete time interval T

$$d^k(u, v) = \min\{C^{(t\cdot\epsilon)}(u, w) + d^{k+\frac{C^{(k\cdot\epsilon)}(u,w)}{\epsilon}}(w, v) : \ (u, w) \in A\} \text{ if } u \neq v, \tag{16}$$

and $d^k(u, u) = 0$ with $k = 0, 1, ..., t_{\max} \cdot \frac{1}{\epsilon}$. Note that the upper index $k + C^{(k\cdot\epsilon)}(u, w) \cdot \frac{1}{\epsilon}$ is always an integer so it falls under the discrete range of $\{0, 1, 2, ...\}$. Equation 16 provides a retrospective way to generate $d^k, k = 0, 1, ..., t_{\max}\frac{1}{\epsilon}$. Since the costs $C^{(t)}(u, v) = C^{(t_{\max})}(u, v), t \geq t_{\max}$ are static with no change for times above $t_{\max}$ we find that the shortest path matrices $d^{(t)}(u, v), t \geq t_{\max}$ are also static. Therefore, we can compute $d^{t_{\max}\cdot\frac{1}{\epsilon}}(u, v) = d^{(t_{\max})}(u, v)$ by using a static computation for the All-to-All shortest paths for the static costs $c^{(t_{\max})}(u, v)$. Starting from the matrices $d^k = d^{t_{\max}\cdot\frac{1}{\epsilon}}, k > t_{\max} \cdot \frac{1}{\epsilon}$ we can apply retroactively Eq. 16 to generate the matrices $d^{t_{\max}\cdot\frac{1}{\epsilon}-1}, d^{t_{\max}\cdot\frac{1}{\epsilon}-2}, ..., d^1, d^0$. We also denote $\sigma^k(u, v) = \sigma^{(k\cdot\epsilon)}(u, v), k = 0, 1, ..., t_{\max} \cdot \frac{1}{\epsilon}$, the discrete series of matrices for the number of shortest paths.

In the following we describe a method to generate both matrices $d^k(u, v), \sigma^k(u, v)$ for the values $k = t_{\max} \cdot \frac{1}{\epsilon}, t_{\max} \cdot \frac{1}{\epsilon} - 1, ..., 1, 0$. Suppose that we have an algorithm that can generate the static matrices $d^{t_{\max}\cdot\frac{1}{\epsilon}}(u, v), \sigma^{t_{\max}\cdot\frac{1}{\epsilon}}(u, v)$ from the static costs $c^{(t_{\max})}(u, v)$. Equation 16 gives a direct way to calculate the values $d^k(u, v), k < t_{\max} \cdot \frac{1}{\epsilon}$. On the other hand, the computation of the values $\sigma^k(u, v), k < t_{\max} \cdot \frac{1}{\epsilon}$ is more difficult and it is based on the following remarks. Firstly, when we have that $d^k(u, v) > C^{(k\cdot\epsilon)}(u, w) + d^{k+C^{(k\cdot\epsilon)}(u,w)\cdot\frac{1}{\epsilon}}(w, v)$ occurs for some arc $(u, w) \in A$ then the dynamic shortest path should be updated to $d^k(u, v) = C^{(k\cdot\epsilon)}(u, w) + d^{k+C^{(k\cdot\epsilon)}(u,w)\cdot\frac{1}{\epsilon}}(w, v)$. It means that the new dynamic shortest path leaves u at the time $t = k \cdot \epsilon$ and then w at the time $t' = k \cdot \epsilon + C^{(k\cdot\epsilon)}(u, w)$ using a dynamic shortest path between w and v. Therefore, the number of shortest paths between u and v at the time $k \cdot \epsilon$ is in fact the number of shortest paths from w to v at the time t' which gives that $\sigma^k(u, v) = \sigma^{k+C^{(k\cdot\epsilon)}(u,w)\cdot\frac{1}{\epsilon}}(w, v)$. Secondly, we consider the case when $d^t(u, v) = C^{(t\cdot\epsilon)}(u, w) + d^{k+C^{(k\cdot\epsilon)}(u,w)\cdot\frac{1}{\epsilon}}(w, v)$. This means that we have some new shortest paths from u to v that go through w so that the value $\sigma^k(u, v))$ should be increased with $\sigma^{k+C^{(k\cdot\epsilon)}(u,w)\cdot\frac{1}{\epsilon}}(w, v)$. Note that the index $k + C^{(k\cdot\epsilon)}(u, w) \cdot \frac{1}{\epsilon}$ is capped to $t_{\max} \cdot \frac{1}{\epsilon}$. The computation of the dynamic values $d^k(u, v), \sigma^k(u, v)$ is illustrated in Algorithm 1.

3.3 Adapted Computation for the Static Values $d(u, v)$ and $\sigma(u, v)$

We now need to detail the function StaticShortestPaths which is to compute both $d(u, v)$ and $\sigma(u, v)$ for a static graph $G = ((V, A), c)$. We present an approach that can be applied to any All-to-All shortest path algorithm to generate both these elements. We chose to illustrate how the All-to-All Dijkstra algorithm can be adapted, however any All-to-All shortest path algorithm can be modified to achieve this.

Algorithm 1 DynamicShortestPaths($\epsilon,\ t_{\max},\ n,\ c,\ d, \sigma$)

Require: $\epsilon,\ t_{\max},\ n,\ c$
Ensure: d, σ
  // the initialisation step
  for $t = 0$ to $t_{\max} \cdot \frac{1}{\epsilon}$ **do**
    for each $u, v \in V$ **do**
      $d^t(u, v) \leftarrow (u == v)?0 : c^{\lceil t \cdot \epsilon \rceil}(u, v)$
      $\sigma^t(u, v) \leftarrow 1$
    end for
  end for
  // find the shortest path at time $t_{\max}$ using a static solution
  call StaticShortestPaths($n, c^{(t_{\max})}, d^{t_{\max} \cdot \frac{1}{\epsilon}}, \sigma^{t_{\max} \cdot \frac{1}{\epsilon}}$)
  // iterate backwards
  for $k = t_{\max} \cdot \frac{1}{\epsilon} - 1$ downto 0 **do**
    for each $u, v \in V$ **do**
      for each $w \in V : (u, w) \in A$ **do**
        if $\left(k + \frac{C^{(k \cdot \epsilon)}(u,w)}{\epsilon} > t_{\max} \cdot \frac{1}{\epsilon}\right)$ **then**
          $k' \leftarrow t_{\max} \cdot \frac{1}{\epsilon}$
        else
          $k' \leftarrow k + \frac{C^{(k \cdot \epsilon)}(u,w)}{\epsilon}$
        end if
        if $\left(d^k(u, v) > C^{(k \cdot \epsilon)}(u, w) + d^{k'}(w, v)\right)$ **then**
          $d^k(u, v) \leftarrow C^{(k \cdot \epsilon)}(u, w) + d^{k'}(w, v)$ and $\sigma^k(u, v) \leftarrow \sigma^{k'}(w, v)$
        else
          if $\left(d^k(u, v) == C^{(k \cdot \epsilon)}(u, w) + d^{k'}(w, v)\right)$ **then**
            $\sigma^k(u, v) \leftarrow \sigma^k(u, v) + \sigma^{k'}(w, v)$
          end if
        end if
      end for
    end for
  end for

The algorithm StaticShortestPaths is based on the All-to-All Dijkstra algorithm with few modifications to achieve the values $\sigma(u, v)$. Firstly, the values $\sigma(u, v)$ are all initialised with 1 to express that between a node s and any node v there is always a shortest path. Then the algorithm uses the Dijkstra computation based on a data structure Q from which we can extract the node u with the smallest value for $d(s, u)$. The relaxation process for the node u is then modified to accommodate the relaxation for the values $\sigma(u, v)$. Firstly, when $d(s, u) + c(u, v) < d(s, v)$ then $d(s, v)$ is no longer the shortest path between s and v. Hence, the value $d(s, v)$ changes to $d(s, u) + c(u, v)$ and the value $\sigma(s, v)$ to $\sigma(s, u)$. This is because a shortest path between s and v is given by a shortest path between s and u plus the arc (u, v). Secondly, when $d(s, u) + c(u, v) = d(s, v)$ occurs then we find that some new shortest paths can be found through the node u so that the value $\sigma(s, v)$ must increase with $\sigma(s, u)$ (see Algorithm 2 for more details).

Algorithm 2 StaticShortestPaths($n,\ c,\ d,\ \sigma$)

Require: $n,\ c$
Ensure: d, σ
  for each node $s \in V$ **do**
    for each node $v \in V$ **do**
      if $(s == v)$ **then**
        $d(s, v) \leftarrow 0$ and $\sigma(s, v) \leftarrow 0$
      else
        $d(s, v) \leftarrow \infty$ and $\sigma(s, v) \leftarrow 1$
      end if
    end for
    $Q \leftarrow V$
    while Q is not empty **do**
      $u \leftarrow$ pop(Q) - extract from Q the node with the smallest $d(s, u)$
      if $(d(s, u) == \infty)$ **then**
        break
      end if
      for each node $v \in V : (u, v) \in A$ **do**
        $alt \leftarrow d(s, u) + c(u, v)$
        if $(alt < d(s, v))$ **then**
          $d(s, v) \leftarrow alt$ and $\sigma(s, v) \leftarrow \sigma(s, u)$
        else
          if $(alt == d(s, v))$ **then**
            $\sigma(s, v) \leftarrow \sigma(s, v) + \sigma(s, u)$
          end if
        end if
      end for
    end while
  end for

4 Dynamic Snapshot Centrality Indices

With the dynamic elements represented by $d^{(t)}(u, v)$, $\sigma^{(t)}(u, v)$, $\sigma^{(t)}(u, w; v)$ and $\sigma^{(t')}(u, w; v) \downarrow t$ we can now present the new dynamic centrality indices. The aim is to define quantitative measures that reflect the importance or workload of nodes at a given snapshot in time.

Definition 2 The dynamic *snapshot* centrality indices are defined for the dynamic network $G^{(t)} = \left(G = (V, A), c^{(t)}\right)$

$$C_S(v) \downarrow t = \sum_{u \neq v \neq w} \sum_{t' \in T^t(u,v)} \sigma^{(t')}(u, w; v) \downarrow t \text{ - stress-snapshot} \tag{17}$$

$$C_B(v) \downarrow t = \sum_{u \neq v \neq w} \sum_{t' \in T^t(u,v)} \frac{\sigma^{(t')}(u, w; v) \downarrow t}{\sigma^{(t')}(u, w)} \text{ - betweenness-snapshot} \tag{18}$$

We can observe that if a shortest path starts from u at the time $t' \in T^t(u, v)$ in the past then it reaches the node v at the current time t. Therefore, these dynamic centrality indices give a snapshot of the importance of nodes based on the amount of traffic or information converging on the node at a specific time t.

Using the results of the previous sections we can compute the dynamic snapshot centrality values in the FIFO case. For example, Eq. 12 provides a way to calculate the values (Eqs. 17 and 18) by using the following equations:

$$C_S(v) \downarrow t = \sum_{u \neq v \neq w} \sum_{t' \in T^t(u,v)} \sigma^{(t')}(u, v) \cdot \sigma^{(t)}(v, w) \tag{19}$$

$$C_B(v) \downarrow t = \sum_{u \neq v \neq w} \sum_{t' \in T^t(u,v)} \frac{\sigma^{(t')}(u, v) \cdot \sigma^{(t)}(v, w)}{\sigma^{(t')}(u, w)} \tag{20}$$

We will construct the indices for each node by stepping forward in time from the earliest time point. We start by initialising all indices $C_*(v) \downarrow t$ to 0. The core step of the algorithm is to note that for a given node pair u and w, if the shortest path from u to w starting at time t' visits v at time t, we update $C_*(v) \downarrow t$. Based on these remarks Eqs. 19 and 20 can generate the following lines of computation

$$C_S(v) \downarrow t' \leftarrow C_S(v) \downarrow t' + \sum_{v \neq w} \sigma^{(t)}(u, v) \cdot \sigma^{(t+d^{(t)}(u,v))}(v, w), \forall u \in V \tag{21}$$

$$C_B(v) \downarrow t' \leftarrow C_B(v) \downarrow t' + \sum_{v \neq w} \frac{\sigma^{(t)}(u, v) \cdot \sigma^{(t+d^{(t)}(u,v))}(v, w)}{\sigma^{(t)}(u, w)}, \forall u \in V \tag{22}$$

which express the fact that the indices $C_S(v) \downarrow t'$ and $C_S(v)) \downarrow t'$ are being updated with $\sum_{v \neq w} \sigma^{(t)}(u, v) \cdot \sigma^{(t+d^{(t)}(u,v))}(v, w)$ and $\sum_{v \neq w} \frac{\sigma^{(t)}(u,v) \cdot \sigma^{(t+d^{(t)}(u,v))}(v,w)}{\sigma^{(t)}(u,w)}$ respectively. However, these computation lines still reflect the continuous case as $t \in [0, \infty)$ so we need to translate them to the discrete case. For that we denote $C_*^k(v) = C_*(v) \downarrow (k \cdot \epsilon)$ and use the series of matrices represented by d^k and σ^k to obtain the following lines, which are used for the dynamic snapshot centrality indices in the discrete case:

$$C_S^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v) \leftarrow C_S^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v) + \sum_{v \neq w} \sigma^k(u, v) \cdot \sigma^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v, w)$$

$$C_B^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v) \leftarrow C_B^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v) + \sum_{v \neq w} \frac{\sigma^k(u, v) \cdot \sigma^{k+d^k(u,v)\cdot\frac{1}{\epsilon}}(v, w)}{\sigma^k(u, w)},$$

for any node $u \in V$ and for any discrete index $k = 0, 1, ..., t_{\max} \cdot \frac{1}{\epsilon}$. Therefore, the computation of the snapshot dynamic indices firstly finds the values $d^k(u, v)$ and $\sigma^k(u, v)$, and then it calculates the snapshot indices directly with the update process described above. In this process the index $k + d^k(u, v) \cdot \frac{1}{\epsilon}$ is capped to $t_{\max} \cdot \frac{1}{\epsilon}$ when $k + d^k(u, v) \cdot \frac{1}{\epsilon} > t_{\max} \cdot \frac{1}{\epsilon}$. Algorithm 3 presents the details of the computation for the snapshot dynamic indices.

Algorithm 3 SnapshotDynamicIndex($\epsilon,\ t_{\max},\ n,\ c, C_B, C_S$)

Require: $\epsilon,\ t_{\max},\ n,\ c$
Ensure: C_C, C_G, C_B, C_S
  // apply DynamicShortestPaths to calculate d, σ
  call DynamicShortestPaths($\epsilon,\ t_{\max},\ n,\ c,\ d, \sigma$)
  // Initialise the values C(v)
  for each node $v \in V$ **do**
    for $k = 0$ to $t_{\max} \cdot \frac{1}{\epsilon}$ **do**
      $C_B^k(v) \leftarrow C_S^k(v) \leftarrow 0$
    end for
  end for
  // Calculate the values C[v]
  for each node $v \in V$ **do**
    for $k = 0$ to $t_{\max} \cdot \frac{1}{\epsilon}$ **do**
      for each node $u \in V :\ u \neq v$ **do**
        if $\left(k + d^k(u, v) \cdot \frac{1}{\epsilon} > t_{\max} \cdot \frac{1}{\epsilon}\right)$ **then**
          $k' \leftarrow t_{\max} \cdot \frac{1}{\epsilon}$
        else
          $k' \leftarrow k + d^k(u, v) \cdot \frac{1}{\epsilon}$
        end if
        for each node $w \in V :\ w \neq v$ **do**
          if $\left(d^k(u, v) + d^{k'}(v, w) == d^k(u, w)\right)$ **then**
            $C_B^{k'} \leftarrow C_B^{k'} + \frac{\sigma^k(u,v) \cdot \sigma^{k'}(v,w)}{\sigma^k(u,w)}$ and $C_S^{k'} \leftarrow C_S^{k'} + \sigma^k(u, v) \cdot \sigma^{k'}(v, w)$
          end if
        end for
      end for
    end for
  end for

4.1 The Complexity of the Snapshot Dynamic Indices Computation

This section presents the complexity for the snapshot dynamic indices scheme, which is illustrated in Algorithm 3. Firstly, the complexity of the function StaticShortestPaths is identical to the complexity of the Dijkstra algorithm which is $O\left(m \cdot n + n^2 \cdot \log n\right)$ when Q is a Fibonacci heap. Secondly, DynamicShortestPaths calls the function StaticShortestPaths and iterates retrospectively the computation of $d^k(u, v),\ \sigma^k(u, v)$ for $k = t_{\max} \cdot \frac{1}{\epsilon} - 1, t_{\max} \cdot \frac{1}{\epsilon} - 2, ..., 0$. Algorithm 1 shows that each retrospective step k requires $O\left(m \cdot n + n^2\right)$ operations for the graph traversal, therefore the complexity of the DynamicShortestPaths function is $O\left(m \cdot n + n^2 \cdot \log n + t_{\max} \cdot \frac{1}{\epsilon} \cdot \left(m \cdot n + n^2\right)\right)$. Finally, the complexity of SnapshotDynamicIndex requires $O\left(t_{\max} \cdot \frac{1}{\epsilon} \cdot n^3\right)$ to calculate all the dynamic centrality values on top of the DynamicShortestPaths complexity. Therefore, we can conclude that the overall complexity of the SnapshotDynamicIndex function is $O\big(m \cdot n + n^2 \cdot \log n + t_{\max} \cdot \frac{1}{\epsilon} \cdot \left(m \cdot n + n^2 + n^3\right)\big)$ which can be reduced to $O\left(t_{\max} \cdot \frac{1}{\epsilon} \cdot n^3\right)$.

Table 1 Running times for snapshot dynamic indices with $t_{\max} = 50, 100$ and 200

n	25	36	49	64	81	100	121	144
m	60	84	112	144	180	220	264	312
$t_{\max} = 50$	0.1	0.5	0.9	1.9	3.6	6.3	11.6	18.2
$t_{\max} = 100$	0.2	0.7	1.3	3.1	6.3	11.4	19.9	34.4
$t_{\max} = 200$	0.3	1.2	3.3	6.2	14.1	21.9	41.7	69.4

5 Evaluation

This section describes the experiments that were conducted to test the performance of the algorithm to compute the dynamic snapshot indices. The experiments consider a grid network of size nr with $n = nr^2$ nodes and $m = 2 \cdot nr \cdot (nr - 1)$ arcs in which each node is connected to its left, right, top and bottom neighbours. The experiments were carried out on a Pentium D CPU 2 × 2.80 GHz, with 1 GB of RAM. Each experiment was run three times and the average of the execution times in seconds is reported.

5.1 Experiment 1: Running Times for Dynamic Indices

Our first experiment investigates how the algorithm performs as we vary the problem size, the time window, and the granularity of the discretization, for FIFO costs. The dynamic network is based on the grid structure described above. For each arc (u, v) in the grid, a static integer value $r(u, v)$ was randomly generated as the base to calculate the dynamic costs. We then generate the dynamic costs for each arc (u, v) using $c^{(t+1)}(u, v) = c^{(t)}(u, v) \cdot (1 + \delta)$, $c^{(0)}(u, v) = r(u, v)$, with $\delta \in (0, 1]$ a random number. The nature of this definition gives dynamic costs $c^t(u, v)$ that are increasing over time so that the FIFO rule is satisfied. Moreover, since δ is real, the dynamic costs are also real with fractional parts.

Case 1 Fixed value for ϵ. We set $\epsilon = 1$ so that all the costs are integer, and we vary the grid size nr and $t_{\max}$. nr takes values from $5, 6, \ldots, 12$ so that the number of nodes is $n = 25, 36, \ldots 144$. $t_{\max}$ takes values 50, 100, 200. The execution times are presented in Table 1.

Case 2 Fixed value for $t_{\max}$. We now fix $t_{\max} = 100$ and vary the graph size and ϵ. The grid size varies as in case 1. ϵ takes values 1, 0.1, 0.01, which means that the costs are first integer and then real with one or two decimal digits. The execution times are presented in Table 2.

Table 2 Running times for snapshot dynamic indices with $\epsilon = 1, 0.1, 0.01$

n	25	36	49	64	81	100	121	144
m	60	84	112	144	180	220	264	312
$\epsilon = 1$	0.2	0.8	1.6	3.6	7.2	13.2	22.4	38.6
$\epsilon = 0.1$	4.1	11.8	30.4	66.8	118.5	235.9	399.7	727.6
$\epsilon = 0.01$	41.3	121.9	317.2	697.1	1254.3	2391.4	4199.5	7871.1

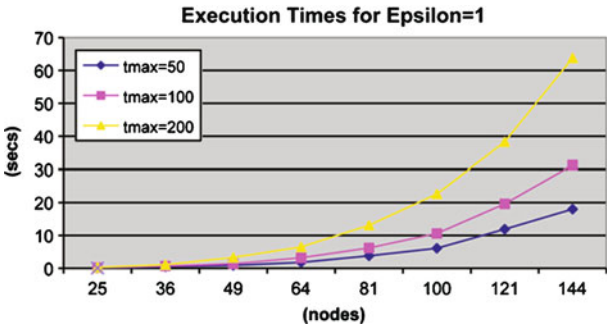

Fig. 1 Running times for snapshot dynamic indices with $t_{\max} = 50, 100$ and 200

The execution times for these two cases are visualised in Figs. 1 and 2. First, we see a linear increase of execution times when the value of $t_{\max}$ increases. Secondly, we can see a sharp increase in the execution times when the value of $\frac{1}{\epsilon}$ increases from 1 to 10 and then to 100. The execution times for $\frac{1}{\epsilon} = 10$ are all between 18 to 22 times bigger than the times for $\frac{1}{\epsilon} = 1$. If we increase the granularity $\frac{1}{\epsilon}$ of the discrete time interval then the execution times can become very large. For example, the execution time exceeds one hour for $\frac{1}{\epsilon} = 100$ and $nr > 11$.

5.2 Experiment 2: Variation of Dynamic Indices

We now consider how the dynamic snapshot betweenness index varies over time in some illustrative problems. The time interval is $t = 0, 1, \ldots, 288$. The dynamic network is based on a grid of size $nr = 11$ in which we consider three different types of dynamic costs $c^{(t)}(u, v)$. The dynamic costs are generated as in the previous experiments from static costs by applying some transformation over time. For this type of grid topology, the closer a node is to the centre of network, the bigger the betweenness value is. Therefore, we expect that the nodes with the biggest static betweenness to be always located amongst the central nodes.

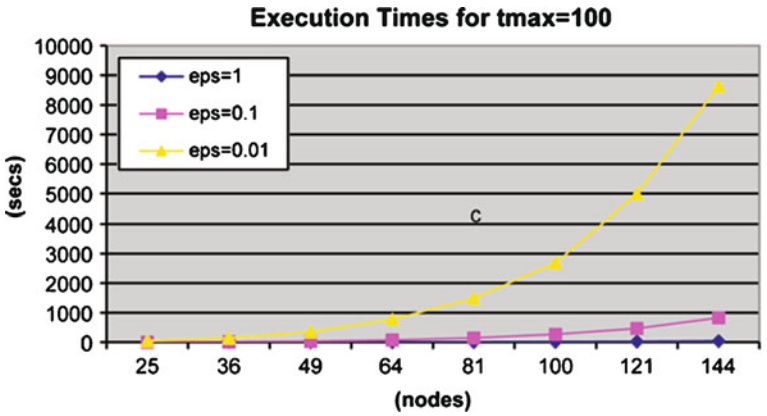

Fig. 2 Running times for snapshot dynamic indices with $\epsilon = 1, 0.1, 0.01$

Case 1 Wave Variation. Firstly, we consider the dynamic costs given by a wave function $c^{(t)}(u,v) = r(u,v) + \frac{r(u,v)}{2} \cdot \sin\left(\frac{\pi \cdot t}{12}\right)$. We generate these values for over a period of 24 hours with 12 readings per hour which gives $t_{\max} = 288$. These costs satisfy the FIFO rule when $c^{(t)}(u,v) < c^{(t+1)}(u,v) + 1$, which is equivalent to $r(u,v) < \frac{1}{2\cdot\sin\left(\frac{\pi\cdot t}{12}+\frac{\pi}{24}\right)\cdot\sin\left(\frac{\pi}{24}\right)} \leq \frac{1}{2\cdot\sin\left(\frac{\pi}{24}\right)} \simeq 3.8306$. Therefore, the static costs are initially randomly generated to double values less than 3.8 to make sure that the FIFO rule is satisfied. For these costs we consider only one decimal so that we work with $\epsilon = 0.1$. The dynamic snapshot betweenness was computed for the central node $(5,5)$ over the discrete interval with 2,880 times. We can observe from Fig. 3 that the variation of this node's dynamic snapshot betweenness inherits the wave behavior from the costs. Moreover, the peaks of the variation are similar with the exception of the first and last ones. We can also see that the node snapshot dynamic betweenness is 0 or close to 0 for the first time intervals, which reflects the fact that no or few shortest paths go through this node at that time. Note that all the snapshot values are 0 for the initial moments.

Case 2 Traffic Function Variation. Secondly, we consider the grid to be a transportation network in which the central nodes are congested during the morning and the afternoon traffic hours. This means that the arcs closer to the centre change dynamically their costs based a traffic distribution with two peaks. This traffic distribution generates the dynamic costs $c^{(t)}(u,v)$ as follows. Firstly, the values $c^{(t)}(u,v)$ are constant equal to the static cost $r(u,v)$ when the time t is outside of two peak time intervals. When t is inside of a time peak interval then the cost $c^{(t)}(u,v)$ increases to a maximum value and then decreases back to $r(u,v)$. We considered the first time peak interval between 8 and 10 am which corresponds to $t = 96, 97, \ldots, 120$ and the second time peak interval between 4 and 8 pm which is for $t = 192, 193, \ldots, 240$. These traffic costs must follow the FIFO rule as overtaking is not permitted in traffic congestion. We restricted the costs to be rational numbers with one decimal place so that $\epsilon = 0.1$. The dynamic snapshot betweenness is shown in Fig. 4 for the nodes (5,5) and (9,4). Note that the node (5,5) is central to the grid therefore it is being used by shortest paths for most of the times outside the peak time intervals. However, during the two peak time intervals, many shortest paths avoid the central nodes so

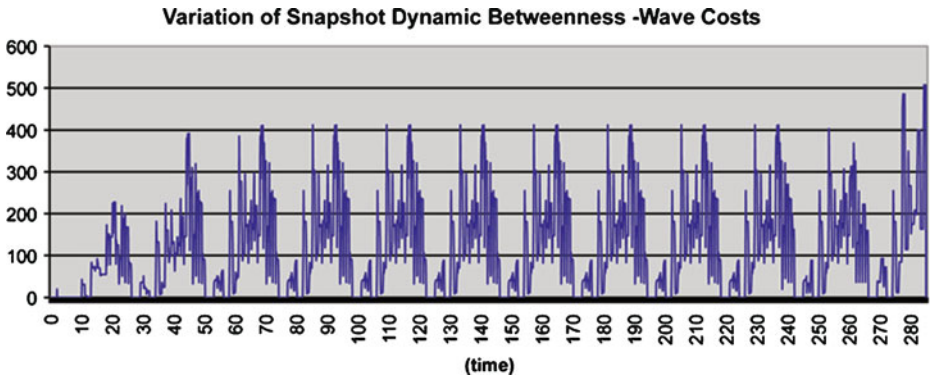

Fig. 3 Variation of the node (5,5)s snapshot dynamic betweenness for wave costs

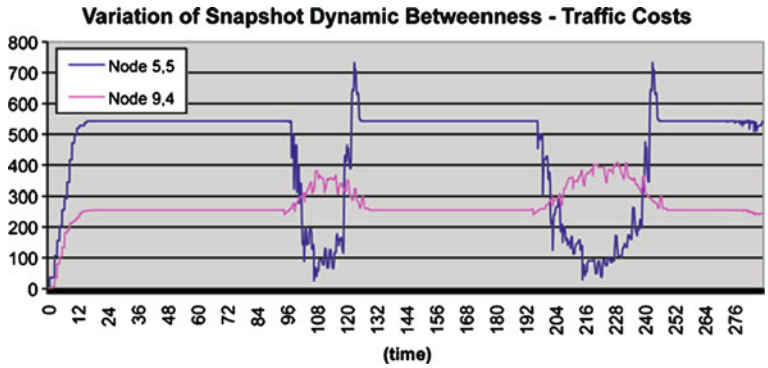

Fig. 4 Variation of the snapshot dynamic betweenness for traffic costs

that (5,5) is visited less often. Hence, we can see the variation of dynamic snapshot betweenness is constant for most times except the two peak intervals. During these two peak intervals, node (5,5) had a sharp decrease in the snapshot betweenness. This loss of snapshot betweenness is however distributed amongst the other nodes of the network as these are now part of more shortest paths. We can observe this behavior on node (9,4), whose snapshot betweenness values increase over the peak time intervals. We can also notice that the node (9,4) becomes more important for traffic than the node (5,5) during the two interval.

Case 3 Variation of Dynamic Indices for Hazard Costs. This experiment considers that the grid network is under some fire hazard. We assume that we know the static navigation costs as estimated times that an able-bodied person could walk the arcs. We assume that the hazard starts at some nodes and then spreads along the arcs through the network with some speed. When a node is caught by the hazard we assume that the adjacent arcs are still usable for a few moments of time then they all become unusable. In this way the navigation costs of the arcs should increase for a few seconds to reflect the hazard and then they become infinity. This model is introduced by Barnes et al. [4] and then generalised by Tabirca et al. [22]. Since

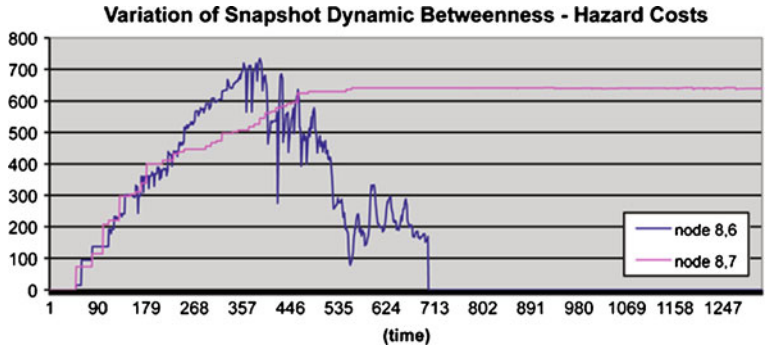

Fig. 5 Variation of snapshot dynamic betweenness for hazard costs

these costs are increasing, they follow the FIFO rule. During the hazard some nodes are caught in the fire, they are unusable for shortest paths and hence their dynamic snapshot betweenness index becomes 0. Because of that the snapshot betweenness values will change over time to reach eventually the value of 0. As before we use the grid network of size 11, with random navigation costs and random speeds for the hazard spread. The fire is then assumed to start at two random locations. Figure 5 shows the variation of the dynamic snapshot betweenness for nodes (8,6) and (8,7). Node (8,6) has the largest snapshot betweenness values for most of the time and then decreases sharply to 0. On the other hand node (8,7) had smaller values initially, but then increases to a constant value over time (because the fire does not spread to this area of the network within the time window).

6 Final Conclusions

The centrality of a node in a graph is a measure of how connected the node is to other nodes, how central it is to a network, or how much information flows through that node as traffic is sent between pairs of nodes. In particular, *stress* and *betweenness* centrality indices assume that traffic is sent along shortest paths, and measures the number of shortest paths that visit individual nodes. However, many practical applications should be modelled by dynamic graphs, in which the structure or the arc traversal times vary over time, and thus the shortest path between any two nodes may change. In such networks, the centrality of a node will change. We therefore considered the idea of *dynamic* centrality for dynamic graphs. In particular, we developed the concept of *snapshot* centrality, which represents the amount of work, or the communication load, being carried out by a node at a single time instant. This allows us to measure the relative importance of a node, or congestion at a node, and how it changes over time.

As is common for dynamic networks, we restricted out attention to first-in, first-out (FIFO) networks, which assume traffic cannot overtake earlier traffic on an arc. Stress and betweenness centrality indices rely on computing shortest paths, and efficient algorithms for this computation require discrete time intervals, but this is restrictive in modelling, where in many cases it is natural to represent changing travel times as continuous functions. We developed a procedure for transforming a continuous valued problem into a discrete valued problem with a variable granularity parameter. We then presented an algorithm for calculating the dynamic indices based on an adapted computation for dynamic shortest paths, which is cubic in the size of the graph, and linear in the granularity of the discretization and in the size of the time window. We demonstrated empirically how this algorithm scales as we varied the granularity, time window and graph size. We then illustrated the dynamic centrality by running experiments on three different cost functions, representing cyclic arc costs, a traffic network with rush hours, and an evacuation problem, where arcs becomes successively harder to traverse as a hazard spreads.

The concept of snapshot centrality arose from application problems in dynamic networks. In future work, we will continue developing the concept for other applications, and we will begin to use the snapshot centrality measures for the underlying design of traffic networks, for improving network reliability, and for the active guidance of transmissions through a network.

References

1. Abraham, A., Hassanien, A.E., Snasel, V.: Computational Social Network Analysis. Springer (2010)
2. Ahuja, R.K., Magnanti, L.T., Orlin, J.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
3. Ahuja, R.K., Orlin, J., Pallottino, S., Scutella, M.G.: Dynamic Shortest Paths Minimizing Travel Times and Costs. MIT Sloan Working Paper No. 4390-02 (2002)
4. Barnes, M., Leather, H., Arvind, D.K.: Emergency evacuation using wireless sensor networks. In: Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN 2007), pp. 851–857 (2007)
5. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **25**, 163–177 (2001)
6. Brandes, U.: On variants of shortest-path betweenness centrality and their generic computation. Soc. Netw. **30**(2), 136–145 (2008)
7. Chabini, I.: A new shortest path algorithm for discrete dynamic networks. In: Proceedings of the 8th IFAC Symposium on Transport Systems, pp. 551–556. Chania, Greece, 16–17 June 1997
8. Chabini, I.: Discrete dynamic shortest path problems in transportation applications: complexity and algorithms with optimal run time. Transp. Res. Rec. **1645**, 170–175 (1998)
9. Cooke, K.L., Halsey, E.: The shortest route through a network with time-dependent internodal transit times. J. Math. Anal. Appl. **14**, 493–498 (1966)
10. Dean, B.C.: Shortest Paths in FIFO Time-dependent Networks: Theory and Algorithms. Technical Report, MIT Department of Computer Science (2004)
11. Demetrescu, C., Italiano, G.F.: Fully dynamic all pairs shortest paths with real edge weights. J. Comput. Syst. Sci. **72**(5), 813–837 (2006)
12. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**, 35–41 (1977)
13. Gross, T., Sayama, H.: Adaptive Networks: Theory, Models and Applications. Springer (2009)
14. Habiba, Tantipathananandh, C., Berger-Wolf, T.Y.: Betweenness Centrality Measure in Dynamic Networks. DIMACS Technical Report 2007-19 (2007)
15. Hage, P., Harary, F.: Eccentricity and centrality in networks. Soc. Netw. **17**, 57–63 (1995)
16. Jacob, R., Koschtzki, D., Lehmann, K.A., Peeters, L., Tenfelde-Podehl, D.: Algorithms for centrality indices. In: Network Analysis, Methodological Foundations, LNCS 3418/2005, pp. 62–82 (2005)
17. Lerman, K., Ghosh, R., Kang, J.H.: Centrality metric for dynamic networks. In: Proceedings of the 8th Workshop on Mining and Learning with Graphs (2010)
18. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**, 026113 (2004)
19. Ramalingam, G., Reps, T.W.: An incremental algorithm for a generalization of the shortest-path problem. J. Algorithms **21**(2), 267–305 (1996)
20. Sabidussi, G.: The centrality index of a graph. Psychometrika **31**, 581–603 (1966)
21. Shimbel, A.: Structural parameters of communication networks. Bull. Math. Biophys. **15**, 501 (1953)
22. Tabirca, T., Brown,K.N., Sreenan, C.J.: A dynamic model for fire emergency evacuation based on wireless sensor networks. In: Proceedings of the 2009 ISPDC Conference, pp. 29–36 (2009)