J Internet Serv Appl (2012) 3:269–275
DOI 10.1007/s13174-012-0073-z

ORIGINAL PAPER

Computing multicast trees in dynamic networks and the complexity of connected components in evolving graphs

Sandeep Bhadra · Afonso Ferreira

Received: 30 August 2011 / Accepted: 8 October 2012 / Published online: 6 November 2012

Abstract Future Internet technologies and the deployment of mobile and nomadic services enable complex communications networks, that have a highly dynamic behavior. This naturally engenders route-discovery problems under changing conditions over these networks, but the temporal variations in the topology of dynamic networks are not effectively captured in a classical graph model. In this paper, we use evolving graphs, which help capture the dynamic characteristics of such networks, in order to compute multicast trees with minimum overall transmission time for a class of wireless mobile dynamic networks. We first show that computing different types of strongly connected components in evolving digraphs is NP-Hard, and then propose a polynomial-time algorithm to build all rooted directed Minimum Spanning Trees in strongly connected dynamic networks. These results open new avenues for the implementation of Internet spanning-tree based protocols over highly dynamic network infrastructures.

Keywords Future internet · Graph theoretic models · Dynamic networks · Wireless networks · Routing · Evolving graphs · Minimum spanning trees

Sandeep Bhadra
Department of Electrical Engineering, Indian Institute of Technology, Madras, Chennai, India
e-mail: sandy@ee.iitm.ernet.in

Afonso Ferreira (✉)
CNRS, I3S, INRIA-Sophia Antipolis, Projet MASCOTTE, BP93, 2004 Rt. des Lucioles, 06902 Sophia Antipolis, France
e-mail: afonso.ds.ferreira@gmail.com

1 Introduction

With the advent of the Internet of Things, it is clear that a globally mobile Internet induces key new challenges related to existing communication and routing solutions [2]. As stated in [15], a major challenge for the Future Internet, where most connected entities will be mobile, is the explicit design of protocols for a mobile wireless world.

Indeed, infrastructure-less mobile communication environments, such as mobile ad hoc networks (MANETs), present a paradigm shift from back-boned networks in that data are transferred from node to node via peer-to-peer interactions and not over an underlying backbone of routers. Naturally, this engenders new problems regarding optimal routing of data under various conditions over these dynamic networks [18].

In this setting, the generalized case of mobile network routing using shortest paths or least cost methods are complicated by the arbitrary movement of the mobile agents thereby leading to random variations in link costs and connectivity [18]. This variable nature of the topology can be apprehended only by network updates of the link state between moving nodes, thus creating substantial communication overhead along the link. This naturally motivates studying the modeling of such dynamics, and designing algorithms that take it into account [19].

Literature related to route discovery issues in dynamic networks started more than four decades ago, with papers dealing with operations of transport networks (e.g., [7,11–13]). Work on time-dependent networks deals with flow algorithms in static networks, with edge traversal times that may depend on the number of flow units traversing it at a given moment. If traversal times are discrete, then the approach proposed in [11], namely of expanding the original graph into T layers representing the time steps (also called space-time

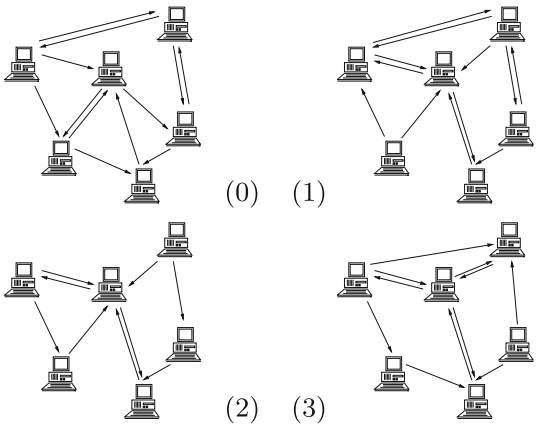

Fig. 1 An FSDN represented as an indexed set of networks.The indices correspond to successive time-steps comp00comp01comp02 comp03

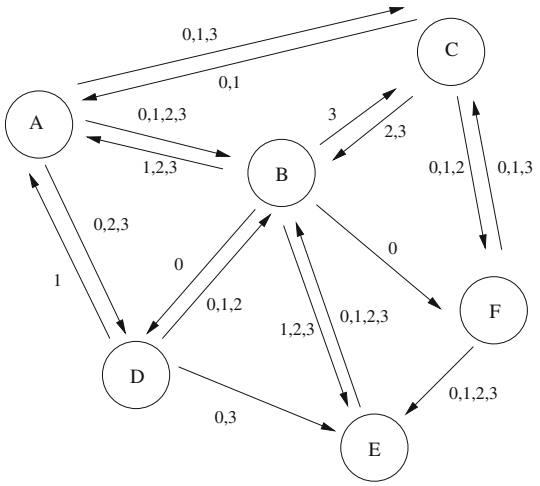

Fig. 2 Evolving digraph corresponding to the FSDN in Fig. 1. Edges are labeled with corresponding time-steps. Observe that CBF is not a valid journey since BF exists only in the past with respect to CB

approach), may work for computing several path-related problems (see [16,17] and references therein). Unfortunately, this approach leads to non-tractable algorithms, since T may be of exponential size.

1.1 Predictable dynamics

Note, however, that for the case of low earth orbiting (LEO) satellite systems, unmanned aerial vehicles (UAV), and other mobile networks with predestined trajectories of the mobile agents, the network dynamics are somewhat deterministic. Therefore, since the trajectories of the network agents are known in advance, it is possible to exploit this determinism in optimizing routing strategies [8,10,20].

Another setting where the evolution of the network is known was studied in [9]. The authors used the notion of competitive analysis [3] on a dynamic setting in order to analyze the quality of a protocol and its online choices made, forced by the evolution of the network. At the end of the process, the *history* of the network is formalized as a sequence of graph topologies on which the application can be solved off-line. The *merit* of the protocol is then the ratio of the solution cost found online over the optimal off-line cost.

Such networks, where the topology dynamics is known or can be predicted beforehand, are henceforth referred to as *fixed schedule dynamic networks* (FSDNs) (see Fig. 1).

1.2 Evolving graphs

Evolving graphs are a formal abstraction for dynamic networks, and can be suited easily to the case of FSDNs [4]. Concisely, an evolving graph is an indexed sequence of $\mathcal{T}$ subgraphs of a given graph, where the subgraph at a given index point corresponds to the network connectivity at the time interval indicated by the index number. The time domain is further incorporated into the model by restricting *journeys* (i.e., the equivalent of paths in usual graphs) to *never* move into edges which existed only in past subgraphs (cf. Fig. 2 below, and Sect. 2). We refer to evolving digraphs to indicate the fact that the edges are directed, implying the same distinction that exists between graphs and digraphs.

Notice that this model allows for arbitrary changes between two consecutive time steps, with the possible creation and/or deletion of any number of vertices and edges. Evolving graph edges can also be associated with traversal times. In [4], algorithms were proposed for finding *foremost, shortest*, and *fastest* journeys in dynamic mobile networks modeled by evolving graphs. Other path problems in evolving graphs can be found under the *merit* approach [9]. Results proven include finding a sequence of paths that connect a given pair of nodes throughout the system, such that the global routing plus re-routing costs are minimized.

1.3 Our work

We focus on the analysis of connectivity properties in FSDNs and the design of algorithms for building directed minimal spanning trees (DMSTs) to generate multicast routes in FSDNs. The DMST problem in wireless networks was defined in [14] as finding N minimum weight trees, or arborescences, in a network modeled by a strongly connected digraph with N vertices. A centralized algorithm for finding DMSTs in static wireless networks is presented by Chu and Liu [5], and Tarjan [21] provides an efficient implementation of the same. Humblet [14] provides a distributed algorithm for finding DMSTs in strongly connected networks. Furthermore, minimum energy multicast trees for wireless networks have also been studied for the static case in [1,22]. In contrast,

our approach differs from these, in that our algorithm builds DMSTs over dynamic mobile networks modeled by evolving digraphs, which can be seen as dynamically changing digraphs.

In this paper, we start by providing, in the next section, basic definitions for various common graph theory terms in the context of evolving digraphs. Following Humblet [14], we define rooted DMSTs over strongly connected evolving digraphs. This naturally leads to the question of how to determine if an evolving digraph is strongly connected. In Sect. 3, we define strongly connected components (SCCs) in evolving digraphs and discover that the unique properties of evolving digraphs yield two types of strongly connected components: standard SCCs and the more loosely defined open strongly connected components (o-SCCs), as it will become clear later. One of our results is that unlike in standard digraphs, finding the strongly connected components in evolving digraphs is not possible in deterministic polynomial time, unless P=NP. In case the evolving digraph is already identified as a strongly connected component, we give in Sect. 4 a polynomial-time algorithm to compute DMST, which uses a variation of Prim's algorithm [6] for computing minimum spanning trees. For an evolving digraph with N nodes and maximum outdegree $\mathcal{D}$, our algorithm builds the rooted DMST over a strongly connected component in an evolving digraph in $O(N\mathcal{D}\log\mathcal{T})$ time. Section 5 contains concluding remarks and scope for further research.

2 Graph theoretic model

Since we use evolving digraphs as a model for FSDNs throughout this paper, we start with a revision of the basic definitions of terms in the theory of evolving digraphs.

2.1 Evolving digraphs

Evolving digraphs are defined as follows.

Definition 1 (*Evolving Digraphs*) Let a digraph $G(V, E)$ be given, along with an ordered sequence of its subdigraphs, $\mathcal{S}_G = G_0, G_1, \ldots, G_{\mathcal{T}}, \mathcal{T} \in \mathbb{N}$. Then, the system $\mathcal{G} = (G, \mathcal{S}_G)$ is called an evolving digraph.

We now define some of the main parameters of an evolving digraph. Let $E_{\mathcal{G}} = \bigcup E_i$, and $V_{\mathcal{G}} = \bigcup V_i$. It is clear that $\mathcal{M} = |E_{\mathcal{G}}| \leq |E| = M$ and that $\mathcal{N} = |V_{\mathcal{G}}| \leq |V| = N$. The central notion in evolving graph theory is the restriction imposed upon paths to traverse arcs strictly in non-decreasing order of arc schedule times, implying that there are no paths in $\mathcal{G}$ going to the "past".

Definition 2 (*Journeys*) Let P be a path in G_i, under the usual definition. Let $F(P)$ be its first vertex, $L(P)$ be its last vertex, and $|P|$ be its length. We define a *journey in* $\mathcal{G}$ between two vertices u and v of $V_{\mathcal{G}}$ as a sequence $\mathcal{J}(u, v) = P_{t_1}, P_{t_2}, \ldots, P_{t_k}$, with $t_1 < t_2 < \cdots < t_k$, such that P_{t_i} is a (usually defined) path in G_{t_i} with $F(P_{t_1}) = u$, $L(P_{t_k}) = v$, and for all $i < k$ it holds that $L(P_{t_i}) = F(P_{t_{i+1}})$.

Corresponding to each arc in $E_{\mathcal{G}}$ we may define an *arc schedule* as a set of indices indicating the presence of the arc in the respective subdigraphs in $\mathcal{S}_G$. Thus, we may alternately define an evolving digraph as a tuple $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$, where each arc in $E_{\mathcal{G}}$ has an arc schedule defined for it.

Two vertices are said to be *adjacent in* $\mathcal{G}$ if and only if they are adjacent in some G_i. The degree of a vertex in $\mathcal{G}$ is defined as its degree in $E_{\mathcal{G}}$.

As usual, a tree in $\mathcal{G}$ could be defined as a connected induced subdigraph of $V_{\mathcal{G}}$ with no circuits in $G(V, E)$. However, such a tree would not be very helpful when studying connectivity issues, since it does not take into account the total order of the subdigraphs in $\mathcal{G}$, and the restrictions it imposes on journeys in $\mathcal{G}$. Therefore, we define a *valid rooted tree in* $\mathcal{G}$ as a rooted directed tree in $\mathcal{G}$, where all paths from the root to the leaves are journeys in $\mathcal{G}$.

2.2 Strongly connected components and arborescences

We define an evolving digraph $\mathcal{G}$ to be a *strongly connected* digraph if there exists a journey $\mathcal{J}$ in $\mathcal{G}$ between any two vertices in $V_{\mathcal{G}}$.

Definition 3 (*Strongly Connected Component*) Analogous to standard digraphs [6], we define a *strongly connected component* (SCC) in an evolving digraph as a maximal set of vertices $U_{\mathcal{G}} \subseteq V_{\mathcal{G}}$ such that for any pair $u, v \in U_{\mathcal{G}}$, there exists a journey from u to v and from v to u using only arcs in the Cartesian product $U_{\mathcal{G}} \otimes U_{\mathcal{G}}$.

Thus, the subdigraph $\mathcal{G}'$ induced by considering vertices in the SCC $U_{\mathcal{G}}$ is a strongly connected digraph. For example, in Fig. 3, $\{b, a\}$ forms a SCC since there are journeys from a to b and vice versa which traverse only vertices in the set $\{a, b\}$. In this figure and elsewhere in the paper arcs are labeled with their respective arc schedule times. Note that, unlike standard digraphs, there can be a journey between two vertices in the SCC that traverses vertices outside $U_{\mathcal{G}}$. Thus, it is possible for two vertices $u, v \in U_{\mathcal{G}}$ to establish a journey between them without the constraint that all arcs in the journey must be within $U_{\mathcal{G}} \otimes U_{\mathcal{G}}$. In Fig. 3, although there exist journeys from b to c and from c to b, $\{b, c\}$ is not an SCC since the only journey from c to b traverses via a. Indeed the subdigraph induced by $\{b, c\}$ is not strongly connected. So, we also offer a looser definition of strong connectivity as follows.

Definition 4 An *open strongly connected component*(o-SCC) is a maximal set of vertices $U_{\mathcal{G}} \subseteq V_{\mathcal{G}}$ such that for any pair $u, v \in U_{\mathcal{G}}$, there exists a journey from u to v and from v to u.

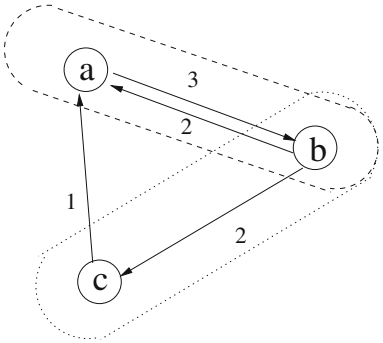

Fig. 3 Open strongly connected components. Arcs are labeled with their respective arc schedule times

A journey between two nodes $u, v \in U_{\mathcal{G}}$, might need to use nodes $h_i \in V_{\mathcal{G}}, h_i \notin U_{\mathcal{G}}$ to maintain strong connectivity. The set of such nodes $\{h_i\} = H(u, v)$ are the helping nodes (*h-nodes*) for the vertices u, v.

Consequently, an SCC $U_{\mathcal{G}}$ is an o-SCC with the additional requirement that $H(u, v) = \emptyset\ \forall u, v \in U_{\mathcal{G}}$. Hence, the set $\{b, c\}$ in Fig. 3 forms a o-SCC with $H(b, c) = \{a\}$ since vertex a is required to form the only journey from b to c, thereby maintaining strong connectivity. Also, since $H(b, c) \neq \emptyset$, $\{b, c\}$ is not an SCC.

For the case of static networks, Humblet [14] defines the concept of rooted spanning trees over strongly connected directed networks. We extend this definition to the case of evolving digraphs as follows. We define a *rooted directed spanning tree* or an *arborescence* over a o-SCC $U_{\mathcal{G}} \in \mathcal{G}$ as a valid rooted directed tree in $\mathcal{G}$ rooted at r which spans all the vertices in $U_{\mathcal{G}}$; thus, all the nodes except the root has one and only one incoming arc. Note that the arborescence might need to include h-nodes to reach some vertices in the o-SCC.

3 Complexity of strongly connected components

In this section we will first use the foremost journey algorithm to verify strong connectivity for an FSDN. Then we will prove that the decomposition of a FSDN into (o-) SCC components is NP-Hard.

3.1 The network model

A FSDN can be seen as a series of networks $\mathcal{R} = \ldots, \mathcal{R}_{t-1}, \mathcal{R}_t, \mathcal{R}_{t+1}, \ldots$ over time. We model a FSDN as a dynamic network which has a *presence* matrix $P_E[(u, v), i]$, indicating whether (u, v) is present at time step t_i, for each link (u, v) of $\mathcal{R}$, and another *presence* matrix $P_V[u, i]$, indicating whether u is present at time step t_i, for each node u of $\mathcal{R}$. The network at time t_i is then represented by the subnetwork $\mathcal{R}_{t_i}$ of $\mathcal{R}$, which is obtained by taking the nodes and links of $\mathcal{R}$ for which their corresponding $P[i]$s indicate they are to be present.

In order to model a fixed-schedule dynamic network by an evolving digraph, it suffices to be given a time window $\mathcal{W}$ of size $\mathcal{T}$, and to work with $\mathcal{G} = (\bigcup \mathcal{R}_i | i \in \mathcal{W}, \text{FSDN}_{|\mathcal{W}})$. Throughout this text, we assume packet-based networks—so transmitting one piece of data equals transmitting one packet over an arc. Link transmission time between nodes in the network may allow for the transmission of a packet over several links before a change in the network topology. Correspondingly in the model, considering time between two successive subdigraphs in an evolving digraph as unity, the time taken to cross an arc (u, v) is expressed as a positive delay $w(u, v) \leq 1$. The case where the traversal time is larger than the frequency of topology change would then yield a delay $w(u, v) > 1$. We also implicitly assume conservation of information, i.e., in case a node in the network disappears for any reason, then upon rejoining the network, it will still have all the information that it had received before its disappearance.

3.2 Verification of strong connectivity in FSDNs

Given an FSDN network, we must determine if it is strongly connected. It is equivalent to the following proposition over the corresponding evolving digraph.

Proposition 1 *Given an evolving digraph $\mathcal{G}$ with $\mathcal{N}$ nodes and $\mathcal{M}$ links over a sequence of length $\mathcal{T}$, it is possible to determine if it is strongly connected or not in $O(\mathcal{N}\mathcal{M}(log\mathcal{T} + log\mathcal{N}))$ time steps.*

Proof The *transitive closure* of $\mathcal{G}$ is defined as the digraph $R_{\mathcal{G}} = (V, E_R)$,where $E_R = \{(v_i, v_j) : \exists$ a journey$\mathcal{J}(v_i, v_j)\}$. Hence, $\mathcal{G}$ is strongly connected if the underlying graph of $R_{\mathcal{G}}$ is a complete graph[1]. The verification is executed simply and efficiently by forming the foremost journeys tree for each node in the network using the algorithm for a single node proposed in [4], whose complexity is $O(\mathcal{M}(log\mathcal{T} + log\mathcal{N}))$. For $\mathcal{N}$ nodes, the algorithm is repeated $\mathcal{N}$ times, for an overall time of $O(\mathcal{N}\mathcal{M}(log\mathcal{T} + log\mathcal{N}))$. □

3.3 Decomposition into SCCs

Tarjan's algorithm [6], based on the concept of *forefathers* in a depth-first search tree over a digraph, is used to decompose standard digraphs into SCCs. However, SCCs in evolving digraphs have the following unique properties, which preclude the use of Tarjan's algorithm.

[1] The underlying graph of $R_{\mathcal{G}}$ refers to the undirected graph that has an edge between v_i and v_j if and only if E_R has both arcs (v_i, v_j) and (v_j, v_i).

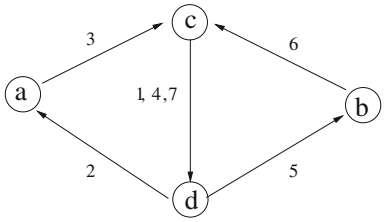

Fig. 4 Overlapping SCCs. Arcs are labeled with their respective arc schedule times

Property 1 *Two different SCCs can have common vertices.*

For example, consider the digraph given in Fig. 4. From the definition of SCCs we see that there are two SCCs a, c, d and b, c, d which have the vertices c, d in common.

Property 2 *For any two vertices in an SCC (respectively, o-SCC), there may be journeys connecting them which use vertices outside the SCC (respectively, o-SCC).*

This stands directly from Property 1. As an example, consider in Fig. 4 the journey from d to c, which uses vertex a that lies outside the SCC $\{b, c, d\}$.

The main problem calls for decomposing the evolving digraph into all possible SCCs. Consider a subproblem *COMPONENT* defined as follows.

COMPONENT: Given an evolving digraph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ and an integer k, is there a SCC of size k?

We shall subsequently demonstrate that *COMPONENT* is NP-Complete, thereby precluding a polynomial time algorithm for the decomposition problem, unless P=NP.

Theorem 1 *COMPONENT is in NP.*

Proof Given a subset $V_{\mathcal{G}'}$ of $V_\mathcal{G}$ and an integer k, we must verify in polynomial time if $V_{\mathcal{G}'}$ is indeed a SCC of size k. First, verifying that $|V_{\mathcal{G}'}| = k$ is easy. Then, verifying that the subdigraph $\mathcal{G}'$ induced by $V_{\mathcal{G}'}$ on $\mathcal{G}$ is strongly connected is possible in polynomial time from Proposition 1. □

We now define a *strong reachability digraph* for an evolving digraph $\mathcal{G}$ as an undirected graph $S_\mathcal{G} = (V_\mathcal{G}, E_S)$, where $E_S = \{(v_i, v_j)\}$ if and only if $(v_i, v_j) \cup (v_j, v_i) \subseteq R_\mathcal{G}$, the transitive closure digraph of $\mathcal{G}$.

To prove the NP-Completeness of *COMPONENT* we reduce the *CLIQUE* problem to *COMPONENT*. *CLIQUE* is formally defined as follows: Given a digraph $G = (V, E)$, and an integer k, is there a clique of size k in G?

Lemma 1 *Finding an SCC in $\mathcal{G}$ is equivalent to finding a maximal clique in $S_\mathcal{G}$, the strong connectivity graph of $\mathcal{G}$.*

Proof Directly from the definitions of strong reachability, SCC and maximal clique, we see that the SCC in $\mathcal{G}$ is equivalent to finding the maximal clique in $S_\mathcal{G}$. □

Theorem 2 *CLIQUE can be reduced to COMPONENT in polynomial time.*

Proof Given an undirected graph $G = (V, E)$ and the integer k, we construct an evolving digraph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ as follows (cf. Fig. 5):

1. For each node $u_i \in V$ create a node $v_i \in V_\mathcal{G}$, a node $h_{ii} \in V_\mathcal{G}$, and arcs (v_i, h_{ii}), (h_{ii}, v_i) with arc schedule time 2;
2. For each edge $\{u_i, u_j\} \in E$, do
   (a) create nodes $h_{ij}, h_{ji} \in V_\mathcal{G}$,
   (b) create arcs (v_i, h_{ij}) and arcs (v_j, h_{ji}), with arc schedule time 2,
   (c) create arcs (h_{ij}, v_j) and arcs (h_{ji}, v_i), with arc schedule time 3.
3. Create an SCC connecting all h-nodes. Label these arcs with schedule times 1 and 4.

By construction of $\mathcal{G}$, its corresponding strong reachability digraph $S_\mathcal{G}$ contains a clique of size formed by all the h-nodes. Let n' denote its size. We can then prove that finding an SCC in $\mathcal{G}$ is the same as finding a clique in G, since a clique of size k in G will correspond to a clique of size $n' + k$ in $S_\mathcal{G}$, corresponding, in turn, to an SCC of size $n' + k$ in $\mathcal{G}$ (via Lemma 1). □

Theorem 3 *COMPONENT is NP-complete.*

Proof We know that *CLIQUE* is NP-Complete. So from Theorems 1 and 2, *COMPONENT* is NP-Complete. □

3.4 Decomposition into o-SCCs

Here, we address the more general result for the case of o-SCC which has a less strict definition than SCC. We define the decision problem as follows.

o-COMPONENT: Given an evolving digraph $\mathcal{G}$ and an integer k, is there a o-SCC of size k?

Although SCCs are a special case of o-SCCs, the NP-Completeness of *COMPONENT* does not directly imply that *o-COMPONENT* is NP-Complete as well. This is because a possible polynomial time algorithm for *o-COMPONENT* need only answer the above decision problem and not identify the o-SCCs of size k, thus making it difficult to verify if at least one o-SCC of size k is an SCC as well (in other words if the set of h-nodes is empty or not for a particular o-SCC of size k). Also, the same digraph $\mathcal{G}$ may contain both an SCC (of indeterminate size) and an o-SCC of size k, so *o-COMPONENT* would always return "yes", ignoring the presence or absence of a SCC of size k, thereby leaving *COMPONENT* unsolved. Conversely, since SCCs are a special case of o-SCCs, proving *o-COMPONENT* to be NP-Complete does not directly imply that *COMPONENT* is NP-Complete as well.

These arguments would entail an independent proof for the NP-Completeness of o-COMPONENT. Fortunately, how-

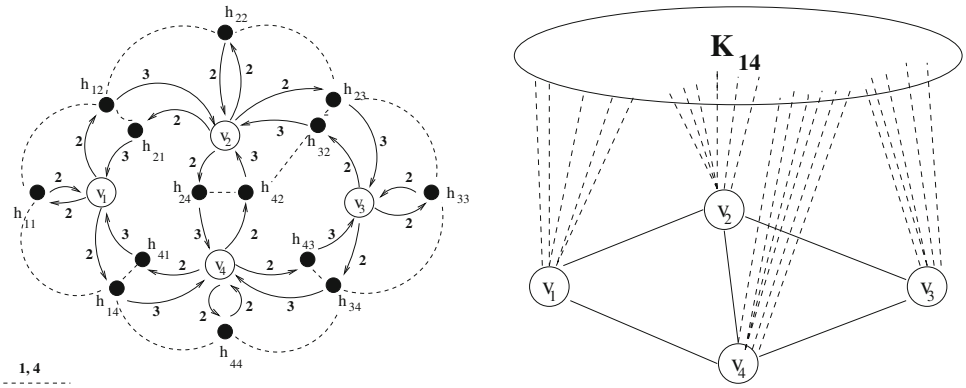

Fig. 5 Construction for Theorem 2. Arcs are labeled with their respective arc schedule times.

ever, the same widget utilized for the previous reduction can be applied in the current case, yielding the following results.

Theorem 4 *o-COMPONENT is in NP.*

Proof The proof of this result is analogous to the one of Theorem 1. It can be derived step by step from the previous proof, but showing it here would be quite fastidious and would not add new insights to this paper. Consequently, we leave the full development of this proof to the interested reader. □

Theorem 5 *CLIQUE can be reduced to o-COMPONENT in polynomial time.*

Proof Given an undirected graph $G = (V, E)$ and the integer $k > 3$, the same arguments used in the proof of Theorem 2 apply here. Indeed, the same widget can be used to reduce *CLIQUE* to o-SCC, since a SCC is a o-SCC where $H = \emptyset$, and in that widget, a max o-SCC is a max SCC, which by Theorem 2 implies the reduction from *CLIQUE*. □

Theorem 6 *o-COMPONENT is NP-complete.*

Proof We know that *CLIQUE* is NP-Complete. So from Theorem 4 and Theorem 5, *o-COMPONENT* is NP-Complete. □

4 Computing directed Minimum Spanning Trees

Considering a strongly connected evolving digraph $\mathcal{G}$, the object is to find $\mathcal{N} = |V_\mathcal{G}|$ rooted directed Minimum Spanning Trees rooted at each of the nodes $r \in V_\mathcal{G}$. Our algorithm is a modification of the Prim–Dijkstra algorithm [6] for finding MSTs in undirected graphs. The algorithm proceeds by building a fragment which is a subset of the DMST starting from the root r. The property of the fragment $f(r)$ is that it consists of those edges by which information transmitted at the beginning of the time interval from the root r will travel in the shortest time to the vertices already included in the fragment. Having defined a fragment as such, it is easy to see how the algorithm for the DMST proceeds. In the following algorithm we choose from among the set of arcs outgoing from the fragment $f(r)$, the arc with the smallest arc schedule time such that it can form a valid journey starting from the root. A number t_v is associated with each vertex $v \in V_\mathcal{G}$ denoting the minimum time required for that vertex to receive the information given that the root r originates the information.

Since each node can transmit information only *after* it has received it, the information cannot pass simultaneously through two edges. Recall that the time required for transmission over one arc is denoted as an arbitrary weight, $w(u, v) < 1$.

In Algorithm 1, an arc schedule time i indicates the presence of the link from time $i - 1$ to i.

Algorithm 1

1. *Start with* $f(r) = \emptyset$ *and a set* V_f *containing vertices already considered in fragment* $f(r)$.
2. $V_f = \{r\}$, $t_r = 1$
3. *while* $V_f \neq V_\mathcal{G}$ *do*
   (a) *Let* Γ_f *be the set of all arcs* (u_i, v_i) *such that* $u_i \in V_f$, $v_i \notin V_f$. *For each* $(u_i, v_i) \in \Gamma_f$, *choose the smallest arc schedule time* $f_a(u_i, v_i)$, *such that* $f_a(u_i, v_i) \geq t_{u_i} + w(u_i, v_i)$.
   (b) *Choose arc* (u_j, v_j) *where* $j = \min_i^{-1}(f_a(u_i, v_i) + w(u_i, v_i))$.
   (c) *if* $f_a(u_j, v_j) = t_{u_j} + w(u_j, v_j)$, *then* $t_{v_j} \leftarrow f_a(u_j, v_j)$,
   (d) *else if* $f_a(u_j, v_j) - 1 \in \{$*arc schedule of* $(u_j, v_j)\}$, *then* $t_{v_j} \leftarrow t_{u_j} + w(u_j, v_j)$,
   (e) *else,* $t_{v_j} \leftarrow f_a(u_j, v_j) - 1 + w(u_j, v_j)$
   (f) *add* v_j *to* V_f *and* (u_j, v_j) *to* $f(r)$.

Note that two cases might arise depending on whether $f_a(u_j, v_j) = t_{u_j} + w(u_j, v_j)$ or $f_a(u_j, v_j) > t_{u_j} + w(u_j, v_j)$. For the first case, the information reaches the node exactly at the time $f_a(u_j, v_j)$. For the other case, if the arc is present both at times $f_a(u_j, v_j) - 1$ and $f_a(u_j, v_j)$, since $w(u_j, v_j) < 1$, the packet will reach v_j in $t_{u_j} + w(u_j, v_j)$. If, however, the arc is not present at time $f_a(u_j, v_j) - 1$, then the transmission process itself starts at the $f_a(u_j, v_j)^{th}$ step (i.e., from time $f_a(u_j, v_j) - 1$ to time $f_a(u_j, v_j)$), thus reaching v_j by time $f_a(u_j, v_j) - 1 + w(u_j, v_j)$.

We remark that a rooted directed tree can also be computed over an o-SCC $V_{\mathcal{G}'}$. As a modification for that purpose,

$V_{\mathcal{G}}$ must be replaced by $V_{\mathcal{G}'}$ and correspondingly, Step 3 of Algorithm 1 should be modified to $V_{\mathcal{G}'} \not\subset V_f$ since the fragment can also contain the h-nodes for the vertices in $V_{\mathcal{G}'}$ and the loop can stop once all the vertices are covered.

Algorithm 1 is a greedy algorithm that always chooses the arc that transmits in minimum time. The proof of its correctness is the same as the proof of the Prim–Dijkstra algorithm [6]. If the maximum outdegree of each vertex is $\mathcal{D}$, then each step of increasing the fragment will take $O(\mathcal{N}\mathcal{D}\log\mathcal{T})$ time and the fragment will increase $\mathcal{N}$ times adding up to a total execution time of $O(\mathcal{N}^2\mathcal{D}\log\mathcal{T})$ steps.

5 Conclusion

The two important results in this paper are the intractability of the decomposition into (open) strongly connected components in FSDNs and the construction of DMSTs over an already existing strongly connected components. Note also that the very concept of Open Connected Components is completely new, and somewhat surprising, arising because of the dynamics of the networks, and may find important applications.

The former result implies that it is possible to lead a non-strongly connected network towards strong connectedness by adding intermediary agents to serve as hops between two nodes that are out of range from each other. An interesting problem would be to find a way to add such links so as to minimize the number of intermediary (helping) nodes. Another way for further research is to design approximation algorithms for (open) strongly connected components in evolving digraphs.

Finally, the construction of minimum spanning trees in networks is part of the solution of many networking problems, like that of finding a low-cost sub-network connecting a set of nodes. We are confident that the foundations laid by this paper will help understand the impact of mobility in networks of the future. The algorithms shown in this paper provide an efficient basis for the deployment of the Future Internet on highly dynamic network environments.

As possible avenues for future work, we know that in order to be more representative of reality, restrictions on network topology changes between consecutive time-slots may be introduced. In the case of mobility, such restrictions could be of geometric nature, e.g., dependent on expected speed and location of nodes. Such modeling of network dynamics could lead to the development of further algorithms, in addition to the general-case and centralized algorithm presented in this paper. Distributed algorithms for this problem also need to be designed.

Acknowledgments The authors are grateful to Aubin Jarry and Stephane Perennes for very fruitful discussions. The authors would also like to thank the reviewers, in particular Reviewers 2 and 3, for the very careful reading of the document. Their comments helped to increase the quality of this paper.

References

1. Ambuehl C (2005) An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: Proceedings of 32th ICALP, pp 1139–1150
2. Andersen FU, Berndt H, Abramowicz H, Tafazolli R (2007) Future internet from mobile and wireless requirements perspective. http://www.emobility.eu.org/
3. Borodin A, El-Yaniv R (1998) Online computation and competitive analysis. Cambridge University Press
4. Bui-Xuan B, Ferreira A, Jarry A (April 2003) Computing shortest, fastest, and foremost journeys in dynamic networks. Int J Found Comput Sci 14(2):267–285
5. Chu YJ, Liu TH (1965) On the shortest arborescence of a directed graph. Sci Sin 14:1396–1400
6. Cormen T, Leiserson C, Rivest R (1990) Introduction to algorithms. The MIT Press, Boston
7. Dreyfus SE (1969) An appraisal of Some Shortest-Path Algorithms. Oper Res 17:269–271
8. Ekici E, Akyildiz IF, Bender MD (2000) Datagram routing algorithm for LEO satellite networks. In: IEEE Infocom, pp 500–508
9. Faragó A, Syrotiuk VR (2001) MERIT: a unified framework for routing protocol assessment in mobile ad hoc networks. In: Proceedings of ACM Mobicom 01, pp 53–60, ACM
10. Ferreira A, Galtier J, Penna P (2002) Topological design, routing and handover in satellite networks. In: Stojmenovic I (ed) Handbook of wireless networks and mobile computing, pp 473–493. Wiley, New York
11. Ford LR, Fulkerson DR (1958) Constructing maximal dynamic flows from static flows. Oper Res 6:419–433
12. Halpern J (1977) Shortest route with time dependent length of edges and limited delay possibilities in nodes. Zeitschrift für Oper Res 21:117–124
13. Halpern J, Priess I (1974) Shortest path with time constraints on movement and parking. Networks 4:241–253
14. Humblet PA (1983) A distributed algorithm for minimum weight directed spanning trees. IEEE Trans Commun COM-31(6):756–762
15. Issarny V, Georgantas N, Hachem S, Zarras A, Vassiliadis P, Autili M, Gerosa MA, Ben Hamida A (2011) Service-oriented middleware for the future Internet: state of the art and research directions. J Internet Serv Appl 2(1):23–45
16. Köhler E, Langkau K, Skutella M (2002) Time-expanded graphs for flow-dependent transit times. In: Proceedings of ESA'02
17. Köhler E, Skutella M (2002) Flows over time with load-dependent transit times. In: Proceedings of the 13th annual ACM-SIAM symposium on discrete algorithms, pp 174–183
18. Scheideler C (2002) Models and techniques for communication in dynamic networks. In: Alt H, Ferreira A (eds) Proceedings of the 19th international symposium on theoretical aspects of computer science, vol 2285. Springer, pp 27–49, March 2002
19. Stojmenovic I (ed) (2002) Handbook of wireless networks and mobile computing. Wiley, New York
20. Syrotiuk V, Colbourn CJ (2003) Routing in mobile aerial networks. In: Proceedings of WiOpt'03—modeling and optimization in mobile, ad-hoc and wireless networks, pp 293–302, INRIA, Sophia Antipolis, March 2003
21. Tarjan RE (1977) Finding optimum branchings. Networks 7:25–35
22. Wieselthier J, Nguyen G, Ephremides A (2000) On the construction of energy-efficient broadcast and multicast trees in wireless networks. In: Proceedings of IEEE Infocom, pp 585–594, Tel Aviv, 2000