Computing multicast trees in dynamic networks using evolving graphs

Sandeep Bhadra, Afonso Ferreira

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computing multicast trees in dynamic networks using evolving graphs

Sandeep Bhadra — Afonso Ferreira

N° 4531

August 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4531--FR+ENG

INRIA
SOPHIA ANTIPOLIS

Computing multicast trees in dynamic networks using evolving graphs*

Sandeep Bhadra† , Afonso Ferreira ‡

Thème 1 — Réseaux et systèmes
Projet Mascotte

Rapport de recherche n° 4531 — August 2002 — 16 pages

Abstract: New technologies and the deployment of mobile and nomadic services are driving the emergence of complex communications networks, that have a highly dynamic behavior. This naturally engenders new route-discovery problems under changing conditions over these networks. Unfortunately, the temporal variations in the network topology are hard to be effectively captured in a classical graph model. In this paper, we use and extend a recently proposed graph theoretic model, which helps capture the evolving characteristic 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 this model is NP-Complete, and then propose an algorithm to build all rooted directed minimum spanning trees in already identified strongly connected components.

Key-words: Wireless networks, mobile networks, multicast, evolving graphs, LEO satellites, minimum spanning trees, strongly connected components, graph theoretic models, NP-Complete.

* This work was partially supported by the European RTN project ARACNE, the European FET project CRESCCO and the COLOR action *Dynamic*. It was done while the first author was visiting the project MASCOTTE, INRIA/CNRS/UNSA, whose support is acknowledged.
† Indian Institute of Technology, Madras.
‡ CNRS, Projet Mascotte, I3S & INRIA Sophia Antipolis.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Calcul d'arbres de multicast dans des réseaux dynamiques grâce aux graphes évolutifs[§]

Résumé : Le progrès des technologies de communication et le déploiement de services mobiles et nomades engendrent l'émergence de réseaux complexes de communication, qui présentent des comportements très dynamiques. Ceux-ci donnent lieu à de nouvelles questions liées au routage, sous les contraintes crées par les changements du réseau. Ces variations dans le temps de la topologie du réseau ne sont pas très bien modélisées par des graphes classiques. Pour ce faire, le modèle des graphes évolutifs a été récemment introduit.

Dans ce rapport, nous utilisons et élargissons ce modèle dans le but de construire des arbres de multicast ayant des temps de transmission minimums, dans une certaine classe de réseaux dynamiques mobiles sans-fil. Nous montrons d'abord que le calcul de différents types de composantes fortement connexes dans des graphes évolutifs est NP-Complet. Dans le cas où l'entrée est déjà une composante fortement connexe, nous donnons un algorithme pour la construction de tous les arbres couvrants enracinés orientés de poids minimum.

Mots-clés : Réseaux sans-fil, réseaux mobiles, multicast, graphes évolutifs, satellites LEO, arbres couvrants de poids minimum, composantes fortement connexes, graphes, NP-Complet.

1 Introduction

Infrastructureless mobile communication environments, such as mobile ad-hoc networks and low earth orbiting (LEO) satellite systems, present a paradigm shift from backboned networks, such as cellular telephony, in that data is transfered 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 networks.

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 [3]. This variable nature of the topology can be incorporated only by network updates of the link state between moving nodes, thus creating substantial communication overhead along the link.

On the other hand, we note that for the case of LEO satellite systems, sensor networks and other mobile networks with predestined trajectories of the mobile agents, the network dynamics are somewhat deterministic. LEO satellite networks [6, 10, 4, 11] in particular, communicate via *inter-satellite links* (ISL's) between satellites that are in range of each other. While ISL's connecting subsequent satellites in the same orbital plane (*intraplanar*) do not vary with time because the satellites move with zero relative angular velocity to each other, *interplanar* ISL's, between satellites in different orbital planes, vary as the satellites move in and out of range of each other. This results in the dynamic topology of the network. However, since the trajectories of the satellites are known ahead of time, it is possible to exploit this determinism in optimizing routing strategies. In fact, the periodicity of the trajectories allows the definition of a system period T_s [10] as the smallest common integer multiple of an orbit period and one Earth day. Thus, the routing algorithm need be computed for only one system period and the same path can be chosen once every T_s.

Our work deals with communication issues in such networks, henceforth referred to as *fixed schedule dynamic networks* (FSDN's), where the topology dynamics at different time-steps can be predicted (see Fig. 1). Note that optimal route discovery problems in networks are equivalent to standard problems such as shortest path trees and minimum spanning trees over the underlying graphs. Literature on routing issues in such networks usually assume limited or no mobility [12, 9] where link-connectivity changes only very gradually and is incorporated by a system of updates to the topology graph with every change. Unfortunately however, the temporal variations in the network topology cannot be effectively captured in a classical graph model.

Recently, *evolving graphs* [5] have been proposed as a formal abstraction for dynamic networks, and can be suited easily to the case of FSDN's. Evolving graphs have been defined over digraphs to capture the essential directivity of data networks and basically aim to formalize a time domain in graphs. Concisely, an evolving graph is an indexed sequence (of length $\mathcal{T}$) of subgraphs of a given graph, where the subgraph at a given index point corresponds to the network connectivity at the time instant indicated by the index number. The time domain is incorporated into the model by restricting paths to *never* move into arcs which existed only in past subgraphs [cf. Fig. 2]. In this model, it is made clear that

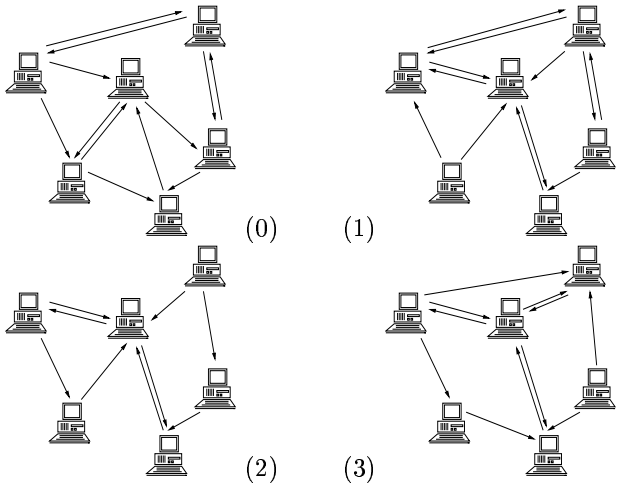

Figure 1: An FSDN represented as an indexed set of networks. The indices correspond to successive time-steps.

between two subsequent time steps, *any* changes may happen, with the possible creation and/or deletion of any number of vertices and arcs. Algorithms for minimum path trees on FSDN's using the evolving graph model have been presented in [5] which compute the shortest path trees in $O(\mathcal{M}(\log \mathcal{T} + \log \mathcal{N}))$ time, for FSDN's corresponding to evolving graphs with $\mathcal{M}$ links and $\mathcal{N}$ nodes.

Presently, our focus is on the analysis of connectivity properties in FSDN's and the design of algorithms for building directed minimal spanning trees (DMST's) to generate multicast routes in FSDN's. The DMST problem in digraphs was defined in [7] as finding N minimum weight trees, or arborescences, in a strongly connected graph with N vertices. A centralized algorithm for finding DMST's in normal digraphs is presented by Chu and Liu [1], and Tarjan [8] provides an efficient implementation of the same. Humblet [7] provides a distributed algorithm for finding DMST's in strongly connected digraphs. Furthermore, minimum energy multicast trees for wireless networks have been studied for the static case in [12, 9]. In contrast our approach differs from these in that our algorithm builds DMST's over evolving graphs, which are dynamically changing digraphs.

In this paper, following Humblet [7], we define rooted DMST's over strongly connected evolving graphs. This naturally leads to the question of how to determine if an evolving graph is strongly connected. We define strongly connected components (SCC's) in evolving graphs and discover that the unique properties of evolving graphs yield two types of strongly connected components: regular SCC's and the more loosely defined open strongly connected components (o-SCC's), as it will become clear later. One of our results is that unlike in regular digraphs, finding the strongly connected components in evolving graphs is not

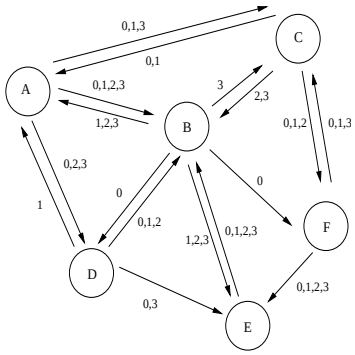

Figure 2: Evolving graph corresponding to FSDN in Fig. 1. Arcs are labeled with corresponding time-steps. Observe that CBF is not a valid path since BF exists only in the past with respect to CB.

possible in deterministic polynomial time, unless P=NP. Finally, we give an algorithm to compute DMST over strongly connected components in evolving graphs, using a variation of Prim's algorithm [2] for MST's. For an evolving graph with maximum outdegree $\mathcal{D}$, our algorithm builds the rooted DMST over a strongly connected component in an evolving graph in $O(\mathcal{N}\mathcal{D}\log\mathcal{T})$ time.

This paper is organized as follows. In the next section we provide basic definitions for various common graph theory terms in the context of evolving graphs. Section 3 contains the algorithm to verify strong connectedness in an evolving graph and the proof of NP-Completeness for SCC's and o-SCC's. The algorithm for computing DMST's in strongly connected components in evolving graphs is presented in Section 4. Section 5 summarizes related work in both the domains of graph algorithms and computing multicast trees for wireless networks. Section 6 contains concluding remarks and scope for further research.

2 Graph Theoretic Model

Since we use evolving graphs as a model for FSDN's throughout this paper, we start with a revision of the basic definitions of terms in the theory of evolving graphs.

2.1 Evolving Graphs

Evolving graphs are defined in [5] as follows.

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

We now define some of the main parameters of an evolving graph. 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 (Paths) *Let P be a path in G_i, under the usual definition. Let $F(P)$ be its source, $L(P)$ be its destination, and $|P|$ be its length. We define a* path in $\mathcal{G}$ *between two vertices u and v of $V_{\mathcal{G}}$ as a sequence $P_{\mathcal{G}}(u,v) = P_{t_1}, P_{t_2}, \ldots, P_{t_k}$, with $t_1 \leq t_2 \leq \cdots \leq t_k$, 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 $L(P_{t_i}) = F(P_{t_{i+1}})$.*

A *circuit in $\mathcal{G}$* is a path in $\mathcal{G}$, $P_{\mathcal{G}}$, such that $L(P_{\mathcal{G}}) = F(P_{\mathcal{G}})$. However, this would imply that each of the subgraphs $G_{t_1}, G_{t_2}, \ldots, G_{t_k}$ must contain the entire circuit. Accordingly, we present a weaker definition in terms of a *cycle* $C_{\mathcal{G}}(u)$ in an evolving graph $\mathcal{G}$, which is defined as a path $P_{\mathcal{G}} = P_{t_1}, P_{t_2}, \ldots, P_{t_k}$ such that $F(P_{t_1}) = u$ and $L(P_{t_k}) = u$ for $t_1 \leq t_2 \leq, \ldots, \leq t_k$.

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 subgraphs in $\mathcal{S}_G$. Thus, we may alternately define an evolving graph 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 $\mathcal{G}$ is defined as a connected induced subgraph of $V_{\mathcal{G}}$ with no circuits in $\mathcal{G}$. However, one such a tree would not be very helpful when studying connectivity issues, since it does not take into account the total order of the subgraphs in $\mathcal{G}$, and the restrictions it imposes on paths in $\mathcal{G}$. Therefore, we define a *valid tree in $\mathcal{G}$* as a tree in $\mathcal{G}$ where each and all directed paths in the tree are paths in $\mathcal{G}$. Likewise, a *valid rooted tree in $\mathcal{G}$* is a rooted directed tree where all paths from the root to the leaves are paths in $\mathcal{G}$.

2.2 Strong Connected Components and Arborescences

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

Definition 3 (Strongly Connected Component) *Analogous to regular graphs [2], we define a* strongly connected component *(SCC) in an evolving graph as the 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 path from u to v and from v to u using only arcs in $U_{\mathcal{G}} \otimes U_{\mathcal{G}}$.*

Thus, the subgraph $\mathcal{G}'$ induced by considering vertices in the SCC $U_{\mathcal{G}}$ is a strongly connected graph. For example, in Fig. 3, $\{b, a\}$ forms a SCC since there are paths 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 regular graphs, there can be a path between two vertices in the SCC that traverses vertices outside

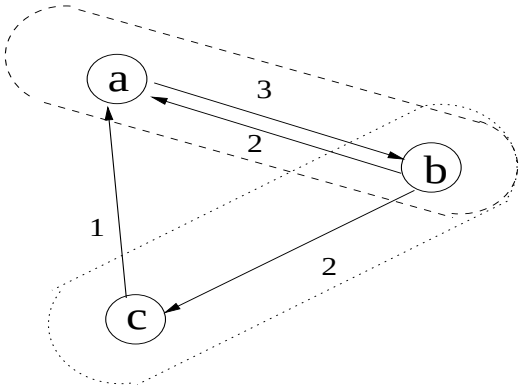

Figure 3: Open Strongly Connected Components.

$U_{\mathcal{G}}$. Thus, it is possible for two vertices $u, v \in U_{\mathcal{G}}$ to establish a path between them without the constraint that all arcs in the path must be within $U_{\mathcal{G}} \otimes U_{\mathcal{G}}$. In Fig. 3, although there exist paths from b to c and from c to b, $\{b, c\}$ is not an SCC since the only path from c to b traverses via a. Indeed the subgraph induced by $\{b, c\}$ is not strongly connected. So, we offer a looser definition of strong connectivity as follows.

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

A path between two nodes $u, v \in U$, might need to use nodes $h_i \in V_{\mathcal{G}}, h_i \notin U$ 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 path 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 [7] defines the concept of rooted spanning trees over strongly connected directed networks. We extend this definition to the case of evolving graphs 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 does not have any cycles and 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 Computing Strong Connected Components in FSDN's

Since directed arborescenses in evolving graphs are defined only over o-SCC's it would be beneficial to decompose the evolving graph corresponding to the FSDN into o-SCC's. In this section we will first present a modification of the shortest path algorithm to verify strong connectivity for an FSDN. Then we will prove that the decomposition of a FSDN into SCC components is NP-Complete. In the next section the more generalized case of an o-SCC is considered, and there too the case is found to be the same, leading to the conclusion that it is necessary to clearly specify the strong connected component before computing the DMST over it.

3.1 The Network Model

We model a FSDN as a series of networks $\mathcal{R} = \ldots, \mathcal{R}_{t-1}, \mathcal{R}_t, \mathcal{R}_{t+1}, \ldots$ over time. An FSDN could be seen 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 (cf. Fig. 2).

In order to model a fixed-schedule dynamic network by an evolving graph, it suffices to be given a time window $\mathcal{W}$ of size $\mathcal{T}$, and to work with $\mathcal{G} = (\bigcup \mathcal{R}_i | i \in \mathcal{W}, \mathrm{FSDN}_{|\mathcal{W}})$. 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 subgraphs in an evolving graph as unity, the time taken to cross an arc (u,v) is expressed as a positive cost $w(u,v) \leq 1$, similar to Scenario 4 of [5]. We also implicitly assume conservation of information, i.e. in case a node in the network fails, then upon rejoining the network, it will retain all the information that it had received before the failure.

3.2 Verification of Strong Connectivity in FSDN's

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

Proposition 1 *Given an evolving graph $\mathcal{G}$ 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 the graph $R_{\mathcal{G}} = (V, E_R)$, where $E_R = \{(v_i, v_j) : \exists P_{\mathcal{G}}(v_i, v_j)\}$. We can now say that $\mathcal{G}$ is strongly connected if $R_{\mathcal{G}}$ is a complete graph. Alternately, the adjacency matrix of the transitive closure $A_{\mathcal{G}}$ must be an all 1's

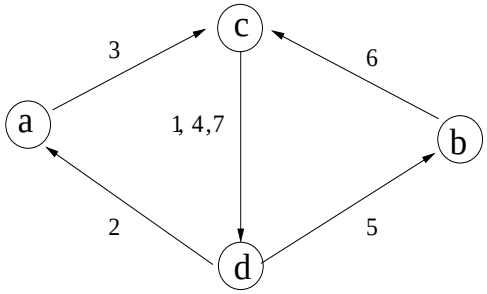

Figure 4: Overlapping SCC's.

matrix. The verification is executed simply and efficiently by forming the shortest paths tree for each node in the network using the algorithm mentioned in [5]. Initialize $A_{\mathcal{G}}$ to the all zeros matrix. For each node the algorithm finds out the minimum distance tree in $O(\mathcal{M}(log\mathcal{T} + log\mathcal{N}))$ operations. If starting from a node u as root, the shortest paths tree on $\mathcal{G}$ contains a node v, then set $A_{\mathcal{G}}(u, v)$ to one. For $\mathcal{N}$ nodes, the algorithm is repeated $\mathcal{N}$ times, taking an overall time of $O(\mathcal{N}\mathcal{M}(log\mathcal{T} + log\mathcal{N}))$. □

3.3 Decomposition into SCC's

Tarjan's algorithm [2], based on the concept of *forefathers* in a depth-first search tree over a graph, is used to decompose regular graphs into SCC's. However SCC's in evolving graphs have the following unique properties, which make it impossible to use Tarjan's algorithm.

Property 1 *Two different SCC's can have common vertices.*

For example, consider the graph given in Fig. 4, where arcs are labeled with the respective arc schedule times. From the definition of SCC's we see that there are two such components a, c, d and b, c, d which have the common vertices c, d between them.

Property 2 *For any two vertices in the SCC (respectively, o-SCC) there may be paths 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 path $d \rightsquigarrow a \rightsquigarrow c$ which uses vertex a that lies outside the SCC $\{b, c, d\}$.

The main problem calls for decomposing the evolving graph into all possible SCC's. Consider a subproblem *COMPONENT* defined as follows.

COMPONENT: Given an evolving graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ 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 the integer k, we must have a means of verifying in polynomial time if $V_{\mathcal{G}'}$ is indeed a SCC of size k. First, verify that $|V_{\mathcal{G}'}| = k$. Next, consider the subgraph $\mathcal{G}'$ induced by $V_{\mathcal{G}'}$ on $\mathcal{G}$ and verify that $\mathcal{G}'$ is strongly connected which is possible in polynomial time from Proposition 1. Thereafter, for each vertex $v \in \mathcal{G}/\mathcal{G}'$, add v to $V_{\mathcal{G}'}$ and add all arcs to $E_{\mathcal{G}'}$ which begin in $V_{\mathcal{G}'}$ and end in v and vice versa. We can now verify using Proposition 1 that this modified graph is not strongly connected. This verifies the maximality of $V_{\mathcal{G}'}$ and completes the verification that $V_{\mathcal{G}'}$ is a SCC of size k. Thus, given an arbitrarily chosen subgraph in $\mathcal{G}$, it is possible to say whether it is SCC in polynomial time. □

We now define a *strong reachability graph* for an evolving graph $\mathcal{G}$ as an undirected graph $S_{\mathcal{G}} = (V_{\mathcal{G}}, E_S)$, where $E_S = \{(v_i, v_j)\}$ if and only if $(v_i, v_j) \cup (v_j, v_i) \in R_{\mathcal{G}}$, the transitive closure graph 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 graph $G = (V, E)$, 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 a graph $G = (V, E)$ and the integer k, we construct an evolving graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ as follows:

1. for each $u_i \in V$ create a $v_i \in V_{\mathcal{G}}$, $\forall\ 1 \leq i \leq |V|, i \in N^+$;
2. for each edge $(u_i, u_j) \in E$, create arcs (v_i, v_j) and (v_j, v_i) in $E_{\mathcal{G}}$ with arcs schedule time $t_{ij} = i + j$. [cf. Fig. 5]

We shall subsequently prove that finding an SCC in $\mathcal{G}$ is the same as finding a clique of the same size in G. From the construction above it can be shown that there exists a strong connection between two vertices v_i and v_j in $\mathcal{G}$ if and only if there is an edge $(u_i, u_j) \in E$. From the construction it is obvious to see that if $(u_i, u_j) \in E$, then $v_i \rightsquigarrow v_j$ and $v_j \rightsquigarrow v_i$, thus making a strong connection between v_i and v_j.

Conversely, consider without loss of generality that $i < j$ and that vertices v_i and v_j are strongly connected even though $(u_i, u_j) \notin E$. It is easy to have a path $v_i \rightsquigarrow v_p \rightsquigarrow v_j$, since $i + p < j + p$. But it is not possible to have any $v_j \rightsquigarrow v_q \rightsquigarrow v_i$ since $j + q > i + q$ and therefore $t_{jq} > t_{iq}$. Thus, the strong reachability graph $S_{\mathcal{G}}$ is isomorphic to G. From

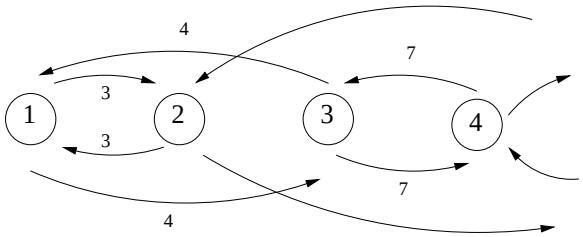

Figure 5: Construction for Theorem 2.

Lemma 1, finding an SCC in $\mathcal{G}$ is equivalent to finding a max-clique in $S_{\mathcal{G}}$ and therefore in G. Thus solving *COMPONENT* over $\mathcal{G}$ solves *CLIQUE* in G. The construction above is in $O(|V| + |E|)$ time steps. Thus we have reduced *CLIQUE* to *COMPONENT* in polynomial time. □

Theorem 3 COMPONENT *is NP-Complete.*

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

3.4 Decomposition into o-SCC's

Here we prove 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 graph $\mathcal{G}$ and an integer $k > 3$, is there a o-SCC of size k?

Although SCC's are a special case of o-SCC's, 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-SCC's 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 graph $\mathcal{G}$ 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. Moreover, since SCC's are a special case of o-SCC's, proving *o-COMPONENT* to be NP-Complete does not directly imply that *COMPONENT* is NP-Complete as well. This entails for an independent proof for the NP-Completeness of o-COMPONENT.

Theorem 4 o-COMPONENT *is in NP.*

Proof Same as the proof for Theorem 1 □

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

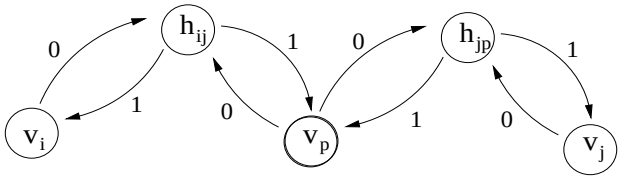

Figure 6: Construction for Theorem 5.

Proof Given an undirected graph $G = (V, E)$ and the integer $k > 3$, we construct an evolving graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ as follows:

1. for each $u_i \in V$ create a $v_i \in V_{\mathcal{G}}$;
2. for each edge $(u_i, u_j) \in E$, do [cf. Fig. 6]
   (a) create a node $h_{ij} \in V_{\mathcal{G}}$,
   (b) create arcs (v_i, h_{ij}),(v_j, h_{ij}) with arc schedule time 0,
   (c) create arcs (h_{ij}, v_i) and (h_{ij}, v_j) with arc schedule time 1.

By the construction, for all $(u_i, u_j) \in E$, $v_i \rightsquigarrow v_j$ and$v_j \rightsquigarrow v_i$ thus making them strongly connected. However any path of the type $v_i \rightsquigarrow v_p \rightsquigarrow v_j$ is not possible due to the design of the arc schedule times (see Fig. 6). Thus for every edge $(u_i, u_j) \in E$, there are edges (v_i, v_j),(v_i, h_{ij}) and (v_j, h_{ij}) in the strong reachability graph $S_{\mathcal{G}}$. However the degree of each h_{ij} is 2 and so they cannot be part of any clique of size greater than 3. Thus from Lemma 1 there is a clique of size $k > 3$ in G, if and only if there is a clique of size k in $S_{\mathcal{G}}$. Thus if *o-COMPONENT* can be solved, *CLIQUE* can be decided for $k > 3$. Hence *CLIQUE* reduces to *o-COMPONENT* for $k > 3$. □

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 the Directed Minimum Spanning Trees

Considering a strongly connected evolving graph $\mathcal{G}$, 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 [2] for finding MST's in undirected regular 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 included already 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 path 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, $w(u,v) < 1$.

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* $f_a(u_i, v_i) \geq t_{u_i} + w(u_i, v_i)$.
   (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* $f(r)$.

In the above algorithm, an arc schedule time i indicates the presence of the link from time $i-1$ to i. 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 $V_{\mathcal{G}'} \subset V_f$ 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 [2]. 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 Related Work

The problem of generating multicast trees in networks and the related problem of finding the minimum spanning trees in directed graphs have been studied in [12, 9, 7, 1, 8]. The centralized algorithm for normal directed graphs, first devised by Chu et al. [1], builds $|V|$ rooted arborescences in $O(|E||V|)$ time steps for a digraph with $|V|$ vertices and $|E|$ edges. In this algorithm, the smallest arc entering each node (apart from the root) is considered as part of the MST. For each cycle formed, the nodes in the cycle are contracted into a pseudo-node (k), and the cost of each arc, which enters a node (j) in the cycle from some node (i) outside the cycle, is modified according to the equation $c(i,k) = c(i,j)+(c(x(j),j)-min_j(c(x(j),j))$ where $c(x(j),j)$ is the cost of the arc in the cycle which enters j. For each pseudo-node, the entering arc which has the smallest modified cost is selected and the arc which enters the same *real* node in the tree is replaced by the new selected arc. The process is repeated until there are no more cycles. Tarjan's implementation of this algorithm reduced the time complexity to $O(|E|\log|V| + |V|^2)$ using special data structures.

Humblet's algorithm [7] is distributed in nature and solves the same problem in the same time-complexity. The nodes are assumed to be processors and communicate with each other to build clusters and identify the minimum weight cluster incoming edge.

In [12], Wieselthier et al. propose to build a multicast tree with transmission energy as node cost parameter. The nodes are assumed to be omnidirectional wireless antennae randomly scattered on a flat surface. The multicast tree is built by considering minimum incremental changes made to the total power requirement (considered as a power law over the radial distance between two nodes) for the fragment, in a way similar to Prim's algorithm. Unfortunately, their approach assumes a static distribution of the nodes. Moreover, their heuristic do not guarantee minimum cost trees for node based costs [12]. On the positive side, Wan et al. [9] used geometric structures of Euclidean MST's to prove approximation ratios for all three heuristics considered in [12].

6 Conclusion

The two important results in this paper are the intractability of the decomposition into (open) strongly connected components in FSDN's and the construction of DMST's over an already existing strongly connected components.

The first result implies that although it is not possible to identify a subset of nodes in the network that is strongly connected, there is a way of quickly determining if a collection of

mobile agents are strongly connected to each other. Thus, it is possible to lead a non-strongly connected network towards strong connectedness by adding links. For the case of wireless networks it would mean the addition of one or more intermediary agents (corresponding to h-nodes in the evolving graph) to serve as hops between two nodes that are out of range from each other. The upper bound on the number of such additional links is obviously $O(\mathcal{N}^2)$. An interesting problem would be to find a way to add such links so as to minimize the number of intermediary (helping) nodes.

Moreover, the framework of evolving graphs could be suited to the case of random variations of link costs and connectivity for a formal analysis of problems in dynamic networks with unpredictable topology changes. Flows and queues over evolving graphs could prove to be another interesting area of study to analyze buffering issues in dynamic networks.

References

[1] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[3] C.Scheideler. Models and techniques for communication in dynamic networks. In In H. Alt and A. Ferreira, editors, *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science*, volume 2285, pages 27–49. Springer-Verlag, March 2002.

[4] E. Ekici, I. F. Akyildiz, and M. D. Bender. Datagram routing algorithm for LEO satellite networks. In *IEEE Infocom*, pages 500–508, 2000.

[5] A. Ferreira. On models and algorithms for dynamic communication networks: The case for evolving graphs. In *Proceedings of* 4^e *rencontres francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL'2002)*, Mèze, France, May 2002. INRIA Press.

[6] A. Ferreira, J. Galtier, and P. Penna. Topological design, routing and handover in satellite networks. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, pages 473–493. John Wiley and Sons, 2002.

[7] P. A. Humblet. A distributed algorithm for minimum weight directed spanning trees. *IEEE Transactions on Communications*, COM-31(6):756–762, 1983.

[8] R. E. Tarjan. Finding optimum branchings. *Networks*, pages 25–35, 1977.

[9] P.-J. Wan, G. Calinescu, X. Li, and O. Frieder. Minimum-energy broadcast routing in static ad hoc wireless networks. In *Proc. IEEE Infocom*, pages 1162–1171, Anchorage, Alaska, 2001.

[10] M. Werner and G. Maral. Traffic flows and dynamic routing in leo intersatellite link networks. In *In Proceedings 5th International Mobile Satellite Conference (IMSC '97)*, Pasadena, California, USA, June 1997.

[11] M. Werner and F. Wauquiez. Capacity dimensioning of ISL networks in broadband LEO satellite systems. In *Sixth International Mobile Satellite Conference : IMSC´99*, pages 334–341, Ottawa, Canada, June 1999.

[12] J. Wieselthier, G. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proc. IEEE Infocom*, pages 585–594, Tel Aviv, 2000.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399