IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-9, Sept 2019 E-ISSN: 2347-2693

An Algorithm to Find the Directed Minimum Spanning Trees

A. Navis Vigilia[1*], J. Suresh Suseela[2]

[1]Reg. No: 7374 Manonmaniam Sundaranar University, Tirunelveli, India
[2]Department of Mathematics, St. John's College, Tirunelveli, India

[*]*Corresponding Author: navis.jk@gmail.com, Tel.: +919916855256*

DOI: https://doi.org/10.26438/ijcse/v7i9.233239 | Available online at: www.ijcseonline.org

Accepted: 12/Sept/2019, Published: 30/Sept/2019

***Abstract*—** New technologies and the deployment of mobile and nomadic services are driving the emergence of complex communications networks that have highly dynamic behaviour. 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 in NP-Complete, and then propose an algorithm to build all rooted directly minimum spanning trees in already identified strongly connected components.

***Keywords*—** Wireless networks, mobile networks, multicast, evolving graphs, LEO satellites, minimum spanning trees, strongly connected components, graph theoretic models, NP-complete.

I. INTRODUCTION

This work deals with communication issues in networks, henceforth referred to as *fixed schedule dynamic networks* (FSDN's), where the topology dynamics at different time-steps can be predicted. 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 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 diagraphs to capture the essential directivity of the data networks and basically aim to formalize a time domain in graphs. Concisely, an evolving graph is an indexed sequence (of length 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 the past subgraphs. In this model, it is made clear that 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 $O(M(\log T + \log N))$ time, for FSDN's corresponding to evolving graphs with M links and 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 diagraphs was defined in [7] as finding N mimimum weight trees, or arborescences, in a strongly connected graph with N vertices. 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 diagraphs. 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 diagraphs.

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 he 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 diagraphs, finding the strongly connected components in evolving graphs is not possible in deterministic polynominal 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 D, our algorithm builds the rooted DMST over strongly connected component in an evolving graph in $O(ND\log T)$ time.

This paper is organised as follows. In the next section we provide basic definitions for various common graph thery 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.

II. RELATED WORK

Let $\mathcal{P}$ be a path in G_t, under the usual definition. Let $\mathcal{F}(\mathcal{P})$ be its **source**, $\mathcal{L}(\mathcal{P})$ 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 $\mathcal{P}_{\mathcal{G}}\ (u,v) = P_{t_i}, P_{t_2}, ..., P_{t_k}, with\ t_1 \le t_2 \le ... \le t_k$, such that P_{t_i}, is a path in G_t 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 $\mathcal{G}$, $\mathcal{P}_{\mathcal{G}}$, such that $\mathcal{L}(\mathcal{P}_{\mathcal{G}}) = \mathcal{F}(\mathcal{P}_{\mathcal{G}})$. However, this would imply that each of the subgraphs $G_{t_1}, G_{t_2}, ..., G_{t_k}$ must contain the entire circuit. Accordingly, We present a weaker definition in terms of **cycle** $C_{\mathcal{G}}(u)$ in an evolving graph $\mathcal{G}$ which is defined as a path $P_G = P_{t_1}, P_{t_2}, ..., P_{t_k}$ such that $F(P_{t_1}) = u\ and\ L(P_{t_k}) = u\ for\ t_1 \le t_2 \le, ..., \le t_k$.

Corresponding to each other arc in $\mathcal{E}_{\mathcal{G}}$ 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 $\mathcal{G} = (\mathcal{V}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$, where each arc in $\mathcal{E}_{\mathcal{G}}$ has an arc schedule defined for it.

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

As usual, a **tree** in $\mathcal{G}$ is defined as connected induced subgraph of $\mathcal{V}_{\mathcal{G}}$ with no circuits 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 to the root the leaves are paths in $\mathcal{G}$.

III. STRONG CONNECTED COMPONENTS AND ARBORESCENCES

We define an evolving graph G to be a strongly connected graph if there exists a path P_G in G between any two vertices in V_G

Strongly connected Component

A strongly connected component (SCC) in an evolving graph is 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 $\mathcal{G}^1$ induced by considering vertices in the SCC $U_{\mathcal{G}}$ is a strongly connected graph. For example, in Fig 1, $\{b,a\}$ 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 $U_{\mathcal{G}}$.

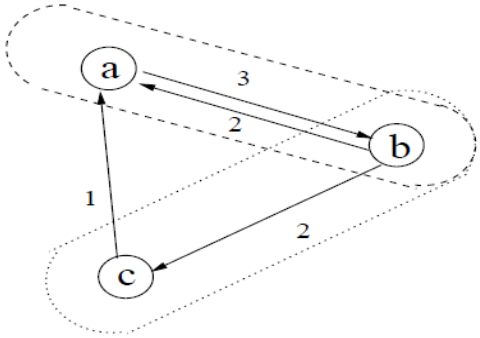

Figure – 1 Open Strongly Connected Component

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 $\mathcal{U}_{\mathcal{G}} \otimes \mathcal{U}_{\mathcal{G}}$. In the fig, although there exist paths from b to c and from c to b, $\{b,c\}$ is not strongly connected.

An open strongly connected component (0-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 $h_i \in V_G, h_i \notin U$ 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_G is an o-SCC with the additional requirement that, $H(u,v) = \phi \forall u,v \in U_G$. Hence the {*b,c*} in Fig 3 forms a 0-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 $H(b,c) \neq \phi, \{b,c\}$ is not an SCC.

IV. COMPUTING STRONG CONNECTED COMPONENTS IN FSDN'S

Since directed arborescences 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.

E NETWORK MODEL

We model a FSDN as a series of networks $R = ..., R_{t-1}, R_t, R_{t+1}, ...$ 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 1time t_i is then represented by the subnetwork R_{t_i} *of* R, which is obtained by taking the nodes and links of $\mathcal{R}$ for which their corresponding $\mathcal{P}[i]$*'s* indicate they are to be present.

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

$$G = \left(UR_i \middle| i \in W, FSDN_{|W}\right)$$

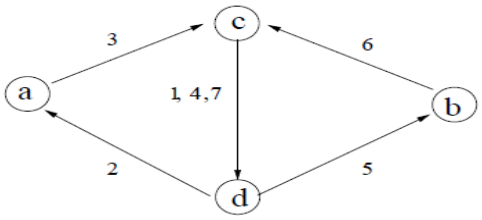

Figure – 2 Overlapping SCC's

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 $w(u,v) \leq 1$. 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.

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 corresponding evolving graph.

Preposition 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 $O(NM(\log T + \log N))$ time steps.

Proof: The transitive closure of $\mathcal{G}$ is defined as 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 $\mathcal{R}_{\mathcal{G}}$ is a complete graph. Alternately, the adjacency matrix of the transitive closure $\mathcal{A}_{\mathcal{G}}$ must be all 1'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 $\mathcal{A}_{\mathcal{G}}$ to the all zeros matrix. For each node the algorithm finds out the minimum distance tree in $O(M(\log T + \log N))$ 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 $O(NM(\log T + \log N))$

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.3, 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 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 property1. As an example consider in Fig.2, the path $d \rightsquigarrow a \rightsquigarrow c$ which uses vertex a that is outside the SCC{*b,c,d*}.
The main problem calls for decomposing the evolving graph into all possible SCC's.

Component: Given an evolving graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ if there is an integer k such that there is SCC of size k.
We shall subsequently demonstrate that COMPONENT is NP-complete, thereby precluding a polynomial time algorithm for the decomposition problem, unless $\mathcal{P}=\mathcal{NP}$.

Theorem 1: COMPONENT is in NP

Proof: Given a subset $V_{\mathcal{G}^1}$ of $V_{\mathcal{G}}$ and the integer k, we must have a means of verifying in polynomial time if $V_{\mathcal{G}^1}$ is indeed a SCC of size k. First, verify that $\left|V_{\mathcal{G}^1}\right| = k$. Next, consider the subgraph $\mathcal{G}^I$ induced by $V_{\mathcal{G}^1}$ on $\mathcal{G}$ and verify $\mathcal{G}^I$ is strongly connected which is possible in polynomial time from Proposition. Thereafter, for each vertex $v \in \mathcal{G}/\mathcal{G}^I$, add v to$V_{\mathcal{G}^1}$ and add all arcs to $E_{\mathcal{G}^1}$ which begin in $V_{\mathcal{G}^1}$ 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 $V_{\mathcal{G}^1}$ and completes the verification that $V_{\mathcal{G}^1}$ 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 $\mathcal{G}$. A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is its induced subgraph is complete.

Lemma1: 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 $\mathcal{S}_{\mathcal{G}}$.

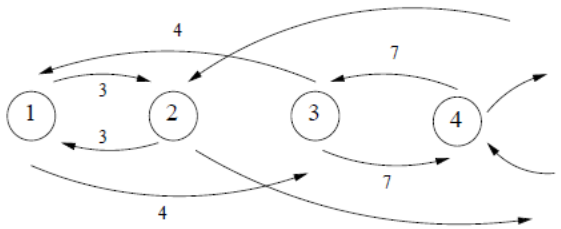

Figure – 3 – Construction for Theorem – 2

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 $v_i \in V_G, \forall 1 \le i \le |V|, i \in N^+$;
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 $t_{ij} = i + j$.

We shall subsequently prove that finding an SCC in $\mathcal{G}$ is the same as finding a clique of 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 $\mathbf{V}_i \rightsquigarrow \mathbf{V}_j$ and $\mathbf{V}_j \rightsquigarrow \mathbf{V}_i$ 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 $\mathbf{V}_i \rightsquigarrow \mathbf{V}_p \rightsquigarrow \mathbf{V}_j$ since $i + p < j + p.$ But it is not possible to have any $\mathbf{V}_j \rightsquigarrow \mathbf{V}_q \rightsquigarrow \mathbf{V}_i$ since $j + q > i + q$ and therefore $t_{jq} > t_{iq}$. Thus, the strong reachability graph $S_{\mathcal{G}}$ is isomorphic to $\mathcal{G}$. From Lemma 1, finding an SCC in $\mathcal{G}$ is equivalent to finding a max-clique in $\mathcal{S}_{\mathcal{G}}$ and therefore in G. Thus solving COMPONENT over G solves CLIQUE in $\mathcal{G}$. The construction above is in $O(|V| + |E|)$ time steps. Thus we have reduced CLIQUE to COMPONENT in polynomial time.

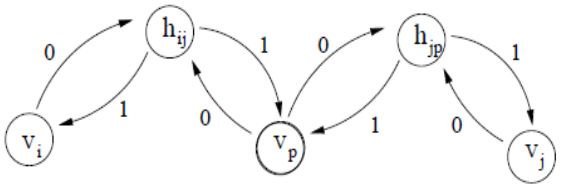

Figure – 4 - Construction for Theorem – 5

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.

DECOMPOSITION INTO O-SCC'S

o-COMPONENT: Given an evolving graph $\mathcal{G}$ and an integer $k>3$, there is 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 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 ofo-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.

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 $v_i \in V_G$;
2. For each edge $(u_i, u_j) \in E$, do
   (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 $(u_i, v_j) \in E$, $v_i \rightsquigarrow V_j$ 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 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 Theorem4 and Theorem 5, o-COMPONENT is NP-Complete.

COMPUTING THE DIRECTED MINIMUM SPANNING TREES

Considering a strongly connected evolving graph $\mathcal{G}$, the object is to find $N = |V_{\mathcal{G}}|$ 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 chose 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 id denoted as an arbitrary weight, $w(u,v) < 1$.

Algorithm

1. Start with $f(r)=\theta$ 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 T_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 $(u_i,v_i)\in T_f$, 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 $f_a(u_j,v_j)=t_u,+w(u_j,v_j)$ 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 $f_a(u_j,v_j)=t_u,+w(u_j,v_j)$ or $f_a(u_j,v_j)>t_u,+w(u_j,v_j)$. 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 $t_{u_i}+w(u_i,v_i)$. 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 $f_a(u_j,v_j)-1+w(u_i,v_i)$.

We remark that a rooted directed tree can also be computed over an o-SCC $V_{\mathcal{G}^1}$. As a modification for that purpose, $V_{\mathcal{G}}$ must be replaced by $V_{\mathcal{G}^1}$ and correspondingly, Step 3 of Algorithm should be modified to $V_{\mathcal{G}^1}\subset V_f$ since the fragment can also contain the h-nodes for the vertices in $V_{\mathcal{G}^1}$ and the loop can stop once all the vertices are covered.

Algorithm 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 N times adding up to a total execution time of $O(\mathcal{N}^2\mathcal{D}\log\mathcal{T})$ steps.

V. CONCLUSION AND FUTURE SCOPE

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 stringly connected, there is a way 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 hinodes in the evolving graph) to serve as hops between two nodes that are out of range from each other.

REFERENCES

[1]. Y. J Chu amd T H. Liu. On the shortest arborescence of a directed graph. Science Sincia, 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 dunamic networks. In In H. Alt and A. Ferreira, editors, Proceedings of the 19th International Symposium on Theoretical Aspecys of Computer Science, volume 2285, pages 27-19. 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]. Ferreira, on models and algorithms for dmanic communication networks: The case for evolving graphs. In Proceedings of 4e rencontres francophnes sur les Aspects Algorithmiques des Telecommunications (ALGOTEL '2002), Meze, France, May 2002.

[6]. Fereira, 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 branching. Networks, pages 25-35, 1977.
[9]. P.-J. Wan, G, Calinescue, 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, pages585-594, Tel Aviv, 2000

Authors Profile

A.Navis Vigilia pursued Bachelor of Science and Master of Science from Sarah Tucker College,Tirunelveli-India in 1990-1995 . She has been working as Lecturer in Department of Mathematical Sciences, Jyoti Nivas College, Bangalore since 2007. She is currently pursuing Ph.D. She has attended many conferences among which

i)1st and 2nd National Conference on Emerging Trends in Fluid Mechanics and Graph Theory in Christ University Bangalore.

ii)National Conference on Mathematical Modeling: A Socio Scientific Approach organised by Tumkur University and Karnataka Higher Education Council

are acknowledgeable. Her area of interests are Graph Theory and Networking.She has published three research papers which are available online. She has 17 years of teaching experience and 6years of Research Experience.

