Time-Aggregated Graphs for Modeling Spatio-temporal Networks*

An Extended Abstract

Betsy George** and Shashi Shekhar

Department of Computer Science and Engineering,
University of Minnesota,
200 Union St SE, Minneapolis, MN 55455, USA
{bgeorge, shekhar}@cs.umn.edu
http://www.cs.umn.edu/research/shashi-group/

Abstract. Given applications such as location based services and the spatio-temporal queries they may pose on a spatial network (eg. road networks), the goal is to develop a simple and expressive model that honors the time dependence of the road network. The model must support the design of efficient algorithms for computing the frequent queries on the network. This problem is challenging due to potentially conflicting requirements of model simplicity and support for efficient algorithms. Time expanded networks which have been used to model dynamic networks employ replication of the network across time instants, resulting in high storage overhead and algorithms that are computationally expensive. In contrast, the proposed time-aggregated graphs do not replicate nodes and edges across time; rather they allow the properties of edges and nodes to be modeled as a time series. Since the model does not replicate the entire graph for every instant of time, it uses less memory and the algorithms for common operations (e.g. connectivity, shortest path) are computationally more efficient than the time expanded networks.

Keywords: Time-aggregated graphs, shortest paths, spatio-temporal data-bases, location based services.

1 Introduction

Growing importance of application domains such as location-based services and evacuation planning highlights the need for efficient modeling of spatio-temporal networks (e.g. road networks) that takes into account changes to the network over time. The model should provide the necessary framework for developing

* This work was supported by the NSF/SEI grant 0431141, Oak Ridge National Laboratory grant and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

** Corresponding author.

J.F. Roddick et al. (Eds.): ER Workshops 2006, LNCS 4231, pp. 85–99, 2006.

efficient algorithms that implement frequent operations posed on such networks. A frequent query that is posed on such networks is to find the shortest route from one place to another or a search for the nearest neighbor. The shortest route would depend on the time dependent properties of the network such as congestion on certain road segments, which would increase the travel time on that segment. The result of nearest neighbor search could also be time sensitive if it is based on a road network.

Modeling such a network poses many challenges. Not only should the model be able to accomodate changes and compute the results consistent with the existing conditions, it should do so accurately and simply. In addition, the need to answer frequent queries quickly means fast algorithms are required for computing the query results. The model should thus provide sufficient support for the design of correct and efficient algorithms for the frequent computations.

Often dynamic networks have been modeled as time expanded networks, where the entire network is replicated for every time instant. The changes in the network, especially the travel time variations, can be very frequent and for modeling such frequent changes, the time expanded networks would require a large number of copies of the original network, thus leading to network sizes that are too memory expensive. For example, traffic sensors on highway networks send measurement data every 30 seconds. A one-year dataset may need over one million copies of the road network, which itself may have a million nodes and edges for each time instant. Such large sized networks would also result in computationally expensive algorithms.

The proposed model, a time-aggregated graph, models the changes in a spatio-temporal network by collecting the node/edge attributes into a set of time series. The model can also account for the changes in the topology of the network. The edges and nodes can disappear from the network during certain instants of time and new nodes and edges can be added. The time-aggregated graph keeps track of these changes through a time series attached to each node and edge that indicates their presence at various instants of time. Our analysis shows that this model is less memory expensive and leads to algorithms that are computationally more efficient than those for the time expanded networks.

1.1 An Illustrative Application Domain

Location based services find the geographical location of a mobile device and then provide services based on that location [11]. Most of these services rely heavily on road maps, spatial networks that can change with time. For example, the travel times associated with road segments can change over time. One of the most frequent computations performed on a road network is to identify the shortest route from one point in the network to another. The result of this query will depend on the availabilty of road segments and the time taken to traverse them; these parameters are time-dependent and hence are the results of the shortest route queries. The results to another frequent query, the nearest neighbor query, can also be time dependent, if computed on a road network; the accessibility of various points in a road network can vary with time, depending on

the connectivity of the network at different instants of time. The need to answer such queries in location based services on a spatial network that varies with time makes a simple, efficient model for spatio-temporal networks a necessity.

Such a model is even more critical to applications related to evacuation planning. Route finding here involves identifying paths in a transportation network to minimize the time needed to move people from disaster-impacted areas to safe locations [6]. One key step in this operation is finding the fastest possible evacuation routes in the network. In computing these routes, it is critical to honor the time dependence of the parameters like travel time (which would change with congestion on roads) and the road capacities. Failure to do so could affect the quality of the solution and even create chaos in an emergency situation.

1.2 Problem Formulation

Spatial networks that show time-dependence serve as the underlying networks for most location based services. Models of these networks need to capture the possible changes in topology and values of network parameters with time and provide the basis for the formulation of computationally efficient and correct algorithms for the frequent computations like shortest paths. We formulate this as the following problem:

Given: The set of frequent queries posed by an application on a spatial network, the pattern of variations of the spatial network with time.
Output: A model which supports efficient and correct algorithms for computating the query results.
Objective: Minimize the storage and computational cost of computation.
Constraints: (1) Edge travel times are positive integers. (2) Edge travel time preserves the FIFO (First-In First-Out) property.

Example: The figures 1(a),(b),(c) show a network at three instants of time. The travel times on the edges (the number shown on the edges) change with time. For example the edge N2-N1 has a travel time of 1 at the instant $t = 1$ and 5 at $t = 2$. It can be seen that the topology of the network is also time-dependent. Though the edge N2-N1 is present at $t = 1$ and at $t = 2$, it is absent at $t = 3$. The task is to develop a model that captures the network across time. The time-aggregated graph for this series of graphs is shown in Figure 1(d).

1.3 Related Work and Our Contribution

Time expanded networks have been widely used to model time dependency of parameters in networks [5,4,9]. This method duplicates the original network for each discrete time unit $t = 0, 1, \ldots, T$ where T represents the extent of the time horizon. The expanded network has edges connecting a node and its copy at the next instant in addition to the edges in the original network, replicated for every time instant. Figure 2 shows an illustration of a time expanded graph. It significantly increases the network size and is very expensive with respect to

memory. Because of the increased problem size due to replication of the network, the computations become expensive.

Stochastic models which use probability distribution functions to describe travel time [4,8,7,3] have been used to study time-dependence of transportation networks. Though they can give valuable insights into the traffic flow analysis, the computational cost to compute the least expected travel times in these networks is prohibitively large to adapt to real life scanarios [8].

Ding [2] proposed a model that addresses the time-dependency by associating a temporal attribute to every edge and node of the network so that its state at any instant of time can be retrieved. This model performs path computations over a snapshot of the network. Since the network can change over the time taken to traverse these paths, this computation might not give realistic solutions.

Our Contribution: In this paper, we propose a model called time-aggregated graphs that represents the time dependence of the network and its parameters. We aggregate the travel times of each edge over the time instants into a time series and keep track of the edges that are present at every instant. We show that this model has less storage requirements than time expanded networks since it does not rely on replication of the entire network across time instants. We also propose algorithms for computing the connectivity and the shortest route from one node to another based on this model. We assume that the travel times of the edges vary with time in a predictable fashion.

1.4 Scope and Outline of the Paper

The main focus of the paper is our proposed use of time-aggregated graphs to represent spatial networks and account for the changes that can take place in the network over a period of time. The model requires that the spatial network be a discrete time dynamic network. The paper proposes algorithms to compute the connectivity and shortest paths in the time-varying network modeled as a time-aggregated graph.

The rest of the paper is organized as follows. Section 2 discusses the basic concepts of the proposed model and provides the relevant definitions of various terms used. Section 3 proposes algorithms for connectivity and shortest path computation based on this model. It also proposes the cost models for these algorithms. In section 4, we conclude and describe the direction of future work.

2 Basic Concepts

Traditionally graphs have been extensively used to model spatial networks [10]; weights assigned to nodes and edges are used to encode additional information. For example, the capacity of a transit station can be represented using the weight assigned to the node that represents the station and the travel time between two stations can be represnted by the weight of the edge connecting the nodes. In a real world scenario, it is not uncommon for these network parameters to be time-dependent. This section discusses a graph based model that can capture

the time-dependence of network parameters. In addition, the model captures the possibility of edges and nodes being absent during certain instants of time.

2.1 A Conceptual Model

A graph $G = (N, E)$ consists of a finite set of nodes N and edges E between the nodes in N. If the pair of nodes that determine the edge is ordered, the graph is directed; if it is not, the graph is undirected. In most cases, additional information is attached to the nodes and the edges. In this section, we discuss how the time dependence of these edge/node parameters are handled in the proposed model, the time-aggregated graph.

We define the time-aggregated graph as follows.
$taG = (N, E, TF, f_1 \dots f_k, g_1 \dots g_l, w_1 \dots w_p | f_i : N \rightarrow \mathbb{R}^{TF}; g_i : E \rightarrow \mathbb{R}^{TF}; w_i : E \rightarrow \mathbb{R}^{TF})$ where
N is the set of nodes, E is the set of edges, TF is the length of the entire time interval, $f_1 \dots f_k$ are the mappings from nodes to the time-series associated with the nodes (for example, the time instants at which the node is present), $g_1 \dots g_l$ are mappings from edges to the time series associated with the edges and $w_1 \dots w_p$ indicate the time dependent weights on the edges.

Definition: Edge/Node Frequency. The number of time steps for which an edge e is present, denoted by $f_E(e)$, is called the frequency of edge e. The edge frequency of a time-aggregated graph f_E is defined as,

$f_E = max_{e \in E_{taG}}(f_E(e))$.

In figure 1, $f_E(N1 - N2) = 2$ and f_E of the time-aggregated graph= 3.

Similarly, we can define $f_N(v)$ to be the number of time steps for which a node v is present and the degree of node presence of a time-aggregated graph is defined as $f_N = max_{v \in N_{taG}}(f_N(v))$

The term 'frequency' used here does not imply periodicity in the edge/node time series.

We assume that each edge travel time has a positive minimum and the presence of an edge at time instant t is valid for the closed interval $[t, t + \sigma]$

Example: Figure 1 shows a time-aggregated graph and the network at each time step. Figures (a), (b) and (c) show the graphs at time instants 1,2 3. (d) shows the time-aggregated graph; each edge has a travel time series (enclosed in square brackets) and the edge time series associated with it. For example, the edge from node N1 to node N2 is present at time instants $1, 2$ and disappears at the time instant $t = 3$. This is encoded in the time-aggregated graph using the edge time series of N1-N2 which is $(1, 2)$; the travel times of this edge for all instants within the time interval under consideration are aggregated into a time series [1,1,-]; the entry '-' indicates that the edge is absent at the time instant $t = 3$.

Figure 2 shows the time aggregated graph (corresponding to Figure 1(a),(b), (c)) and the time expanded graph that represent the same scenario. The time expansion for the example network needs to go through 7 steps since the latest time instant would end in the network is at $t = 7$. For example, the traversal

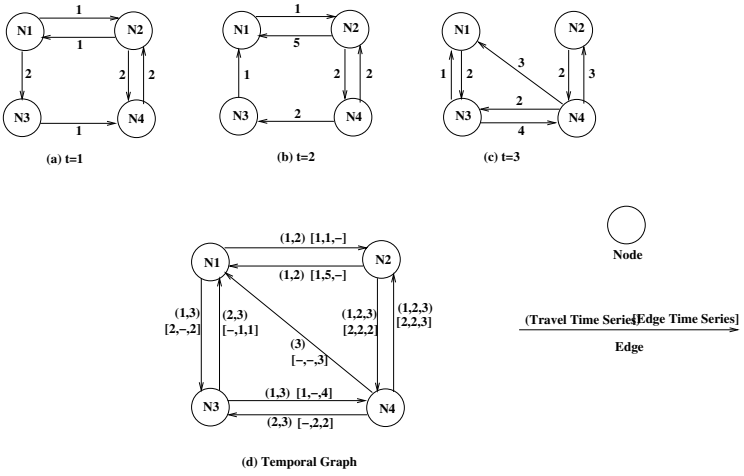


Fig. 1. Time-aggregated Graph, Network at various instants

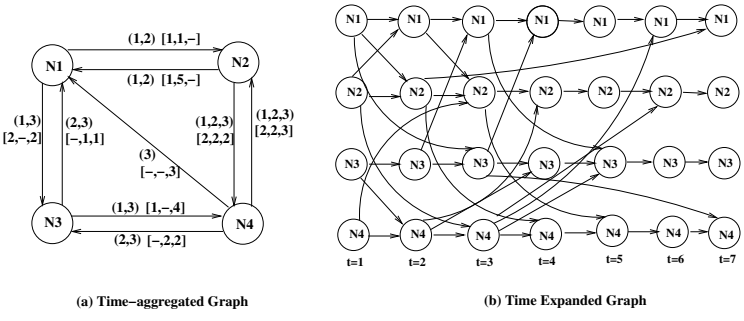


Fig. 2. Time-aggregated Graph vs. Time Expanded Graph

of the edge N3-N4 that starts at $t = 3$ ends at $t = 7$, the travel time of the edge being 4 units. The number of nodes is larger by a factor of T, where T is the number of time instants and the number of edges is also larger in number compared to the time-aggregated graph. Typically the value of T is very large in a spatial network such as a road map since the changes in the network are quite frequent. This would result in time expanded networks that are enormously large and would make the computations slow.

2.2 A Logical Data Model

Basic Graph Operations

We extend the logical data model described in [10] to incorporate the time dependence of the graph model. The three fundamental classes in the model

are **Graph**, **Node** and **Edge**. The common operations that are associated with each class are listed.

```
public class Graph   {
public void add(Object label, timestamp t);
// node with the given label is added at the time instant t.

public void addEdge(Object n1, Object n2, Object label,
timestamp t, timestamp t_time)
// an edge is added with start node n1 and end node n2 at
// time instant t and travel time, t_time.

public Object delete(Object label, timestamp t)
// removes a node at time t and returns its label.

public Object deleteEdge(Object n1, Object n2, timestamp t)
// deletes the edge from node n1 to node n2 at t.

public Object get(Object label, timestamp t)
// returns the label of the node if it exists at time t.

public Iterator get_node_Presence_Series(Object n1)
// the  presence series of  node n1 is returned.

public Object getEdge(Object n1, Object n2, timestamp t)
// returns the edge from node n1 to node 2 at time instant t.

public Iterator get_edge_Presence_Series(Object n1, Object n2)
// the  presence series of edge from node n1 to node n2
// is returned.

public Object get_a_Successor_node(Object label, timestamp t)
//  an adjacent node of the vertex is returned if an edge exists
//  to this node at a time instant at or after t.

public Iterator get_all_Successor_nodes(Object label, timestamp t)
//  all adjacent nodes are returned if edges exist to them
//  at time instants at or after t.

public Object get_an_earliest_Successor_node(Object label,timestamp t)
//  the adjacent node which is connected to the given node with
//  the earliest time stamp after t is returned.

public timestamp get_node_earliest_Presence(Object n1,
                                        timestamp t)
//  the earliest time stamp after t at which the node n1
//  is available is returned.

public timestamp get_node_Presence_after_t(Object n1,
                                       timestamp t)
```

```
//  Part of the presence time series of node n1 after time t
//  is returned.


public timestamp get_edge_earliest_Presence(Object n1, Object n2,
                                                    timestamp t)
//  the earliest time stamp after t at which the edge from
// node n1 to node n2 is available is returned.

public timestamp get_edge_Presence_after_t(Object n1, Object n2,
                                                   timestamp t)
//  Part of the presence time series of edge(n1-n2) after time t
//  is returned.

}
```

A few important operations associated with the classes **Nodes** and **Edges** are provided below.

```
public class Node  {
public Node(Object label, timestamp t)
// the constructor for the class. A node with the appropriate
// label is created at the time t.

public Object label()
// returns the label associated with the node if it exists at t.
}

public class Edge  {
public Edge(Object n1, Object n2, Object label,
timestamp t_inst, timestamp t)
// the constructor for the class. an edge is added with start
// node n1 and end node n2 at time instant t and
// travel time, t_time.

public Object start()
// returns the start node of the edge.

public Object end()
// returns the end node of the edge.
}
```

The table 1 shows the difference in the behavior of the logical operators when the temporal dimension is removed.

We also define two predicates on the time-aggregated graph.

exists_at_time_t: This predicate checks whether the entity exists at the start time instant t.

exists_after_time_t: This predicate checks whether the entity exists at a time instant after t.

Table 1. Logical operators with and without 'time' dimension

Operator	with Time	without Time
delete	deletes the node for the specified time instant	deletes the node for the entire time period
deleteEdge	deletes the edge for the specified time instant	deletes the edge for the entire time period
get	get(node,time)	get_node_Presence_series(node)
getEdge	getEdge(node1,node2,time)	get_edge_Presence_series(node)
get_node_Presence	get_node_earliest_Presence(node, time)	get_node_Presence_series(node)
get_edge_Presence	get_edge_earliest_Presence(node1, node2,time)	get_edge_Presence_series(node, node2)

These predicates are used in conjunction with the entities node, edge and route in a graph to extend the classical graph theory concepts of adjacency, route and connectivity to time-aggregated graphs. Table 2 illustrates these concepts. For example, node v is adjacent to node u at any time t if and only if the edge (u, v) exists at time t as shown in the table. Similarly, a valid route exists from node u to node v if a path exists from node u to node v for start time t at node u and in accordance with the presence of edges along the route.

Table 2. Adjacency, Route and Connectivity in Time-aggregated Graphs

	exists_at_time_t	exists_after_time_t
Node	exists(node u,at_time_t)	exists(node u,after_time_t)
Edge	adjacent(node u,node v, at_time_t)	adjacent(node u,node v, after_time_t)
Route	route(node u,node v,a_route r at_time_t)	route(node u,node v,a_route r, after_time_t)

2.3 Physical Data Model

The adjacency-list and the adjacency-matrix representations are the most common main-memory data structures used in the implementation of graphs. To implement our model, the time-aggregated graphs, a modified version of adjacency list representation is used. This data structure uses an array of pointers, one pointer for each node. The pointer for each node points to a list of immediate neighbors in the time-aggregated graph. At each neighbor node, the edge presence series and travel times for the edge starting from the first node to this neighbor are also stored.

Comparison of Storage Costs with Time Expanded Networks
According to the analysis in [12], the memory requirement for time expanded network is $O(nT) + O(n + mT)$, where n is the number of nodes and m is the

number of edges in the original graph. The memory requirement for the time-aggregated graphs would be $O(m.f_E + n.f_N) + O(n+m)T$, where f_E is the edge frequency and f_N is the node frequency of the time-aggregated graph. This can be simplified to $O(m+n)T$ since T is always greater than f_E and f_N. This comparison shows that the memory usage of time-aggregated graphs is less than time expanded graphs by a factor of $O(nT)$.

The algorithms for connectivity and shortest path computation in time-aggregated network will be discussed in Section 3.

3 Algorithms for Network Computations

The critical step in most queries on a spatio-temporal network is the shortest path computation. Thus the proposed model must include an efficient shortest path algorithm. Here, the shortest path is the route that can be traversed in the shortest time, given the start time at the start node. We assume that there is no cost for waiting at the nodes other than the wait time, and that edge presence is closed for $[t, t+\sigma]$ where σ is the travel time of the edge at the time instant t.

3.1 Algorithm for Shortest Path Computation

As noted earlier, any route computation in a spatio-temporal network must be consistent with the edge presence. Here, the application of a greedy strategy (which is a popular choice in most of the optimization problems) faces a challenge. Not all shortest paths display the optimal sub-structure, which is an essential condition for greedy algorithms to generate an optimal solution. This is clearly illustrated in Figure 3. Although it can be verified that the route, $s-N1-N2-N4-u-d$ is an optimal path from s to d, the route does not display optimal sub-structure since the route from s to u following the above path is not optimal (shortest path being, $s-N1-N3-N5-u$). Though such paths that do not display optimal sub-structure could exist, it can be proved that there is at least one optimal path which satisfies the optimal sub-structure property.

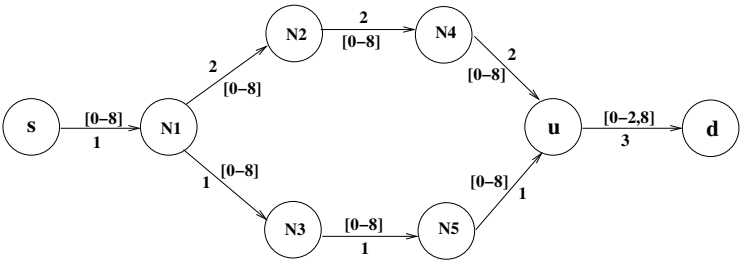


Fig. 3. Illustration of Shortest Paths

Lemma 1. If there is an optimal route from s to d, then there is at least one optimal route from s to d that shows optimal sub-structure.

Proof. As Figure 3 illustrates, the failure of optimal structure of the shortest path occurs due to a potential wait at the intermediate node (u), after reaching this node traversing the optimal path from s to u. Consider the optimal path from s to u. Append this path to the path $u-d$ (allowing wait at the intermediate node u) from the optimal path. This would be still the shortest path from s to d. Otherwise, it would contradict the optimality of the original shortest path.

Lemma 1 enables us to use a greedy approach to compute the shortest path. The algorithm stores the current shortest path travel times to reach every node from the source node. It closes the node with the minimum travel time and updates the travel times of its neighbors. At this step, it uses the operator *get_edge_earliest_Presence* to find the earliest time instant after the arrival at which this edge can be traversed. The updated travel time thus checks for the edge presence and adds the wait time to the travel time, if necessary. The algorithm is similar to Dijkstra's shortest path algorithm, the key difference being the step that looks for the earliest availability of the edges to the adjacent nodes.

Computational Complexity: The cost model analysis assumes an adjacency list representation of the graph with two significant modifications. The edge time series is stored in the sorted order. Attached to every adjacent node in the linked list are the edge time series and the travel time series.

For every node extracted from the priority queue Q, there is one edge time series look up and a priority queue update for each of its adjacent nodes. The time complexity of this step is $O(\log f_E + \log n)$. The asymptotic complexity of the algorithm would be

$$O(\Sigma_{v \in N}[degree(v).(\log f_E + \log n]) = O(m(\log f_E + \log n)).$$

The time complexity of the shortest path algorithm based on a time expanded network is $O(nT \log T + mT)$ [1]. It can be seen that the algorithm based on a time-aggregated graph is faster if $\log n < T \log T$.

3.2 Algorithm for Connectivity

The existence of a valid route from one node to another in a time-aggrgated graph is a non-trivial issue since a path in the time-aggrgated graph does not always guarantee the existence of a path that is consistent with the edge time series and edge travel times. Figure 4 illustrates this; the node N2 is connected to node N4 for starting time instants $1, 2, 3, 4$ (one route being N2 - N5 - N4), and N4 is not accessible from N2 for all time instants after $T = 4$.

Computational Complexity: The cost model analysis assumes an adjacency list representation of the graph with two significant modifications. For every node, the node presence time series is stored in the sorted order. Attached to every adjacent node in the linked list are the edge presence series and the travel time series.

Algorithm 1. Computation of Shortest Path

Input:
1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E;
define type p positive integer
Each node $n \in N$ has two properties:
$NodePresenceTimeSeries$: series of p
Each edge $e \in E$ has two properties:
$EdgePresenceTimeSeries$,$Travel\_time(e)series$: series of p
$\sigma_{u,v}(t)$ is the travel time of edge (u, v) at time t
2) s: Source node, $s \subseteq N_G$;
3) d: Destination node, $d \subseteq N_G$;

Output: Shortest Route from s to d

Method:

```
        c[s] = 0; ∀v(≠ s), c[v] = ∞;
        Insert s in priority queue Q.
   while Q is not empty do {
        u = extract_min(Q);
        v = get_all_Successor_nodes(u, c[u])
            t = get_earliest_Presence(u, v, c[u]);
            if t + σ_{u,v}(t) < c[v] {
                update c[v].
                parent[v] = u;
                if v is not in Q
                  insert v in Q;
            }
        update Q;
   }
Output the route from s to d.
```

For each node dequeued form the queue Q, there is one edge series look up an enqueue operation for each of its adjacent node. The queue operations are $O(1)$ operations. The time complexity of this step is $O(\log f_E)$. The asymptotic complexity of the algorithm would be $O(\Sigma_{v \in N}[degree(v).(\log f_E)]) = O(m(\log f_E))$.

The time dependency of the network parameters affects the connectivity and the shortest paths between nodes in the network. Figure 5 depicts the connectivity and shortest path travel times for different start time instants at the source node for the example network shown in figure 4. Figure 5(a) illustrates the connectivity of the node N2 to node N4 at instants $1, 2, 3, 4, 5, 6$ (these time instants denote the starting times at the node N2). It can be seen that valid routes exist from node N2 to node N4 if the traversal starts at time instants $1, 2, 3, 4$ and that the node N4 is unreachable from N2 for time instants $5, 6$. It might also be interesting to note that the routes that connect the nodes also change with time. For example, at time instant 1, routes N2-N3-N4, N2-N5-N4 and N2-N3-N5-N4 connect N2 to node N4; at starting time, $t = 4$, only N2-N5-N4 is available.

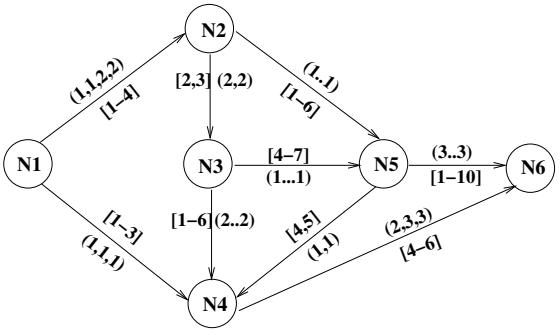


Fig. 4. Illustration of Connectivity

Algorithm 2. Connectivity Algorithm

Input:

1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E;
   define type p positive integer
   Each node $n \in N$ has two properties:
   $NodePresenceTimeSeries$: series of p
   Each edge $e \in E$ has two properties:
   $EdgePresenceTimeSeries$,$Travel\_time(e)series$: series of p
   $\sigma_{u,v}(t)$ is the travel time of edge (u,v) at time t
2) s: Source node, $s \subseteq N_G$;
3) d: Destination node, $d \subseteq N_G$;

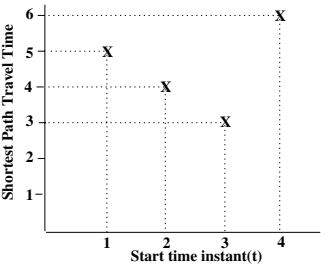
Output: A route from s to d, if exists; else returns FALSE.

Method:

```
Initilialization ;
        Add s to Q; found = false;
        for each node v ∈ N_G do {
                arr_time[v] = 0;
        }
while found = FALSE or Q not empty do {
        u = dequeue(Q);
        v = get_all_Successor_nodes(u, arr_time[v])
                Add v to the Q;
                t = get_earliest_presence(u, v, c[u]);
                if t ≠ ∞ {
                        arr_time[v] = t + σ_{u,v}(t)
                        parent[v] = u
                }
                if v = d, FOUND = TRUE;
                }
}
Output the route from s to d.
```

Start time	1	2	3	4	5	6
Connectivity (N2 to N4)	Yes	Yes	Yes	No	No	No

(a) Connectivity



(b) Shortest Path

Fig. 5. Time Dependency of Connectivity and Shortest Paths

As shown in figure 5(b), the shortest path routes and the travel times are also dependent on time.Consider the shortest path from node N1 to node N6. The shortest path from node N1 to N6, for starting time $t = 1$ is N1-N4-N6 and travel takes 5 units of time (reaches the destination node at $t = 6$). The route remains the same for start times $t = 2, 3$, but the travel time changes to 4 units and 3 units respectively. At time $t = 4$, the route N1-N4-N6 is no longer available and the shortest route changes to N1-N2-N5-N6 with a total travel time of 6 units. This shows that the shortest paths in a time-dependent network vary with time.

4 Conclusions and Future Work

In this paper, we proposed a new model, time-aggregated graphs to model spatio-temporal networks which accounts for the changes in the network topology and parameters with time. Existing approaches rely on time-expanded networks which lead to high storage overhead and computationally expensive algorithms. Time-aggregated graph which models the time dependence using an aggregation of network parameters across the time horizon without the need to replicate the entire graph. We provided algorithms to compute connectivity and fastest route in the network, two frequent types of queries posed on a spatio-temporal network. Our analysis show that this model is less memory expensive compared to time expanded networks and leads to computationally efficient algorithms.

We plan to implement the work and test its performance on road networks. We are working towards extending the model to incorporate turn restrictions which are mostly time dependent and can significantly influence the fastest route computation. We understand that the model should accomodate time-varying capacities of the road networks while performing path computations, especially in applications like evacuation planning where capacity constraints in the network are the key challenge. Though the presence/absence of nodes has not been exploited in the algorithms presented in this paper, we expect that this feature would be used in applications like evacuation planning where nodes might

become unavailable for certain periods of time due to capacity constraints. Also, we need to formulate an algorithm to compute the fastest path in the network over the entire time period or for a user-defined time interval.

Acknowledgment

We are particularly grateful to the members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and valuable discussions. We would also like to express our thanks to Kim Koffolt for improving the readability of this paper.

This work was supported by the NSF/SEI grant 0431141, Oak Ridge National Laboratory grant and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

References

1. B.C. Dean. Algorithms for minimum-cost paths in time-dependent networks. *Networks*, 44, August 2004.
2. Z. Ding and R.H. Guting. Modeling temporally variable transportation networks. *Proc. 16th Intl. Conf. on Database Systems for Advanced Applications*, pages 154–168, 2004.
3. R. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20:182–188, 1986.
4. D.E. Kaufman and R.L. Smith. Fastest paths in time-dependent networks for intelligent vehicle highway systems applications. *IVHS Journal*, 1(1):1–11, 1993.
5. E. Kohler, K. Langtau, and Skutella M. Time-expanded graphs for flow-dependent transit times. *Proc. 10th Annual European Symposium on Algorithms*, pages 599–611, 2002.
6. Q. Lu, B. George, and S. Shekhar. Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. *Proc. of 9th International Symposium on Spatial and Temporal Databases (SSTD'05)*, August 2005.
7. E. Miller-Hooks and H.S. Mahmassani. Least possible time paths in stochastic time-varying networks. *Computers and Operations Research*, 25(12):1107–1125, 1998.
8. E. Miller-Hooks and H.S. Mahmassani. Path comparisons for a priori and time-adaptive decisions in stochastic, time-varying networks. *European Journal of Operational Research*, 146:67–82, 2003.
9. S. Pallottino and M. G. Scuttella. Shortest path algorithms in tranportation models: Classical and innovative aspects. *Equilibrium and Advanced transportation Modelling (Kluwer)*, pages 245–281, 1998.
10. Shekhar S. and Chawla S. *Spatial Databases: Tour*. Prentice Hall, 2003.
11. Shekhar S., Chawla S., R Vatsavai, X. Ma, and Yoo J.S. *Location Based Services, Editors:Schiller, J. and Voisard, A.* Morgan Kaufmann, 2004.
12. D. Sawitzki. Implicit Maximization of Flows over Time . *Technical report, University of Dortmund*, 2004.