Spatio-temporal Network Databases and Routing Algorithms: A Summary of Results

Betsy George*, Sangho Kim, and Shashi Shekhar

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

Abstract. Spatio-temporal networks are spatial networks whose topology and parameters change with time. These networks are important due to many critical applications such as emergency traffic planning and route finding services and there is an immediate need for models that support the design of efficient algorithms for computing the frequent queries on such networks. 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, 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 those for time expanded networks. One important query on spatio-temporal networks is the computation of shortest paths. Shortest paths can be computed either for a given start time or to find the start time and the path that leads to least travel time journeys (best start time journeys). Developing efficient algorithms for computing shortest paths in a time varying spatial network is challenging because these journeys do not always display greedy property or optimal substructure, making techniques like dynamic programming inapplicable. In this paper, we propose algorithms for shortest path computations in both contexts. We present the analytical cost models for the algorithms and provide an experimental comparison of performance with existing algorithms.

Keywords: time-aggregated graphs, shortest paths, spatio-temporal data bases.

1 Introduction

The underlying data of interest for many significant applications such as transportation networks is structured as a spatio-temporal network, which consists

* Corresponding author.

D. Papadias, D. Zhang, and G. Kollios (Eds.): SSTD 2007, LNCS 4605, pp. 460–477, 2007.

of a finite collection of points (i.e. nodes) with location information, the line-segments (i.e. edges) connecting the points, and the time-varying attributes attached to the elements. For example, a spatio-temporal network database for a traveler's trip planning may store the intersections as nodes, the road segments as edges, and time dependent travel time attached to the road segments. In the case of evacuation planning, time dependent capacity may be added to the road segments as another important attribute.

Related work in the field of databases falls into three broad categories (1)Spatial network databases, (2) Graph Databases, and (3) Spatio-temporal databases. The recent release of Oracle (version 10g) includes a network data model to store and maintain the connectivity of link-node networks and supports basic features such as shortest path [14]. The Network Analyst extension of ArcMap from ESRI supports a network geodatabase and provides basic algorithms (e.g., shortest path, service area, closest facility, etc.) [7]. However, these products do not address the time variance of spatial networks, which is crucial in applications such as route computations and emergency planning.

Graph databases [5,6,7,19,22,24] also primarily deal with spatial networks that do not vary with time. Research in graph databases that accounts for temporal variations perform computations over a snapshot of the network [4,9,18], and does not consider the interplay between the edge travel times and the existence of edges. For example, Ding [4] 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. The model does not propose an algorithm for the least travel time paths.

Although the need for live traffic information is increasing, there has been little work on the modeling and algorithms for spatio-temporal network databases. Chorochronos [12], studied various aspects of spatio-temporal databases including ontology, modeling, and implementation. However, the researchers have yet to study spatio-temporal networks in this framework.

Research in Operations Research is based on the time expanded network [10,11,13,15,17,21]. This model 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. This 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.

As the first step towards the study of spatio-temporal network databases, we previously proposed a spatio-temporal network model named the time aggregated graph [8]. In this paper, we introduce a case study of this model using routing algorithms. The proposed algorithms (SP-TAG and BEST) compute the shortest path in the given network for a given start time at the source node and

the least travel time route over the entire time period. The proposed model and algorithms are evaluated with a real world static graph appended with a synthetically generated travel time series.

1.1 An Illustrative Application Domain

Transportation networks are the kernel framework of many advanced transportation systems such as the Advanced Traveler Information System and Intelligent Vehicle Highway Systems. Transportation networks are spatio-temporal in nature and require significant database support to handle the storage of their large amounts of multi-dimensional data. Many important applications based on transportation networks, including travelers' trip planning, consumer business logistics, and evacuation planning need to be built upon spatio-temporal network databases. For example, commuters try to find a suitable time to start their commute so that they spend the least time in traffic. Figure 1 illustrates traffic sensor networks on urban highways which measure congestion levels at two different times (e.g. 5:07pm and 9:37pm) illustrating possible changes in shortest route travel times at different times of the day. With the increasing use of sensor networks to monitor traffic data on spatial networks and the subsequent availability of time-varying traffic data, it becomes important to incorporate this data in the models and algorithms related to transportation networks. However, existing spatio-temporal databases do not offer adequate support for spatio-temporal networks.

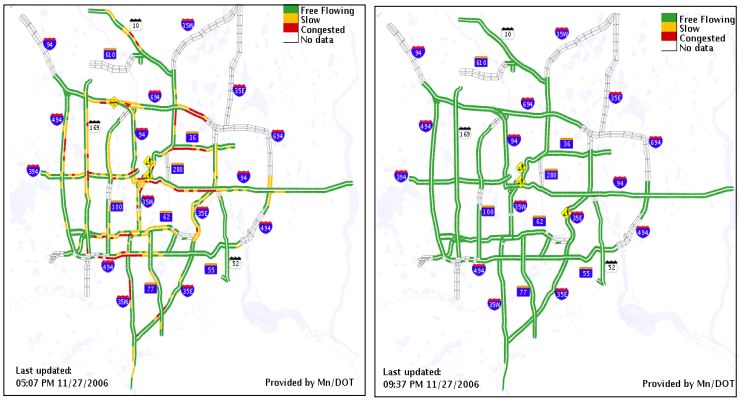

Fig. 1. Sensor networks periodically report time-variant traffic volumes on Twin Cities highways (Best viewed in color, Source: Mn/DOT)

The problem of finding best start time has similar applications in freight delivery services, one of whose main concerns is to reduce logistic costs such as fuel consumption. Another important application is in emergency traffic management. Emergencies caused by natural or manmade disasters can result in atypical

demands on a transportation network, resulting in severe congestion. Emergency managers may be interested in using spatio-temporal network databases to understand non-equilibrium traffic dynamics and to make informed decisions about evacuation route planning.

1.2 Broad Challenges

A time-variant graph is a graph whose edge and node properties and topological structure are time dependent. For example, traffic volume on urban highways varies over the time of day, which leads to a variation in travel time. In addition to network parameter values, the network topology can also change with time due to the unavailability of certain road segments during some periods of time due to repair or natural calamities. Conventional graph algorithms cannot easily be applied to the snapshot graphs at discrete time instants to evaluate frequent queries without accounting for relationships among snapshots. However, time-variant graphs raise many challenges for database research. Due to their potentially large and evergrowing sizes, a storage-efficient representation is critical to reduce and possibly eliminate redundant information across different time-points. Second, new data model concepts need to be investigated to represent and classify potentially new alternative semantics for common graph operations such as shortest-path and connectivity. For example, a shortest path between a given pair of nodes may have at least two interpretations, one for a given start time-point and the other for the shortest travel-time for any start time in a given time interval. A third challenge is the design of efficient and correct query processing strategies and algorithms since some of the commonly assumed graph-properties may not hold for spatio-temporal graphs. For example, consider the optimal substructure (required in dynamic programming, [2]) for shortest paths in a graph. While each prefix path (path from a source node to an intermediate node in an optimal path) is optimal in a static graph, it may not be optimal in a spatio-temporal graph due to a potential wait at an intermediate node.

Our Contribution: The paper describes a model for spatio-temporal networks called the time aggregated graph, that uses a time series to represent time-varying attributes. We propose algorithms to compute shortest paths for a fixed start time and the best start time (Best Start Time Algorithm) and consequently the least commute time paths. These problems are challenging since common algorithm design techniques like greedy design cannot always be applied. The Best Start Time algorithm uses a node cost time series instead of a scalar node cost. The entries in the time series are updated when a path of smaller cost is found. The algorithm iterates until every entry reaches a minimum value and hence does not depend on the greedy choice property. This removes the FIFO restriction from the edge travel times. We also present the experimental analysis of the best start time algorithm and the shortest path algorithm for a given start time [8].

1.3 Scope and Outline of the Paper

The paper presents a case study of time aggregated graphs using routing algorithms to compute shortest paths in two different contexts. Shortest paths can be computed from a given source node for a fixed start time and at the best start time which minimizes the travel time over the entire time horizon.

The rest of the paper is organized as follows. For the sake of completeness, Section 2 provides a brief description of the time aggregated graph model that is used to represent spatio-temporal networks. This section also describes the shortest path algorithm for a given start time. Section 3 describes the proposed algorithm to compute the best start time at a given source node for any destination node. In Section 4, we present the experimental design and the performance analysis. In Section 5 we conclude and describe the direction of future work.

2 Basic Concepts

Spatial networks that show time-dependence serve as the underlying networks for many applications such as routing in transportation networks. Traditionally graphs have been extensively used to model spatial networks (e.g. road networks) [19]; weights assigned to nodes and edges are used to encode additional information. In a real world scenario, it is not uncommon for these network parameters to be time-dependent. It is important to be able to formulate computationally efficient and correct algorithms for the shortest path computation that take into account the dynamic nature of the networks. 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.

Given a set of frequent queries posed by an application on a spatial network and the pattern of variations of the spatial network with time, we need to find a model that supports efficient and correct algorithms for computing the query results, while trying to minimize the storage and cost of computation. In this section we discuss the basics of the model used to represent time dependent spatial networks called "Time Aggregated Networks" [8]. The algorithms presented in this paper are formulated based on this model. Time aggregated graphs can not only capture the time-dependence of network parameters, but also account for the possibility of edges and nodes being absent during certain instants of time.

2.1 The 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 determines 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 edges. In this section, we discuss how

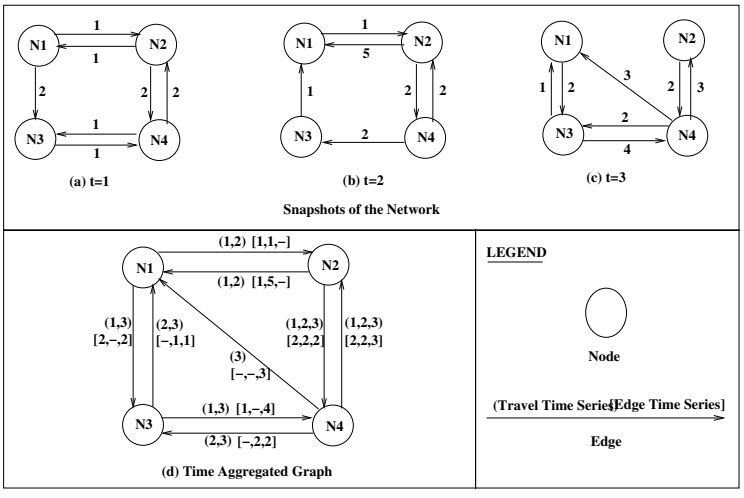

Fig. 2. Network at Various Time Instants and the Time Aggregated Graph

the time dependence of these edge/node parameters are handled in the proposed time-aggregated graph model.

We define the time-aggregated graph as follows.

$taG = (N, E, TF, f_1 \ldots f_k, g_1 \ldots g_l, w_1 \ldots w_p | f_i : N \rightarrow \mathbb{R}^{TF}; g_i : E \rightarrow \mathbb{R}^{TF}; w_i : E \rightarrow \mathbb{R}^{TF})$

where N is the set of nodes, E is the set of edges, TF is the length of the entire time interval, $f_1 \ldots f_k$ are the mappings from nodes to the time-series associated with the nodes, $g_1 \ldots g_l$ are mappings from edges to the time series associated with the edges, and $w_1 \ldots w_p$ indicate the time dependent weights (eg. travel times) on the edges.

Each edge has an attribute, called an edge time series that represents the time instants for which the edge is present. This enables the time aggregated graph to model the topological changes of the network with time. 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 $[t, t + \sigma]$.

Figure 2(a,b,c) shows a network at three time instants. The network topology and parameters change over time. For example, the edge N2-N1 is present at time instants $t = 1, 2$, and disappears at $t = 3$, and its weight changes from 1 at $t = 1$ to 5 at $t = 2$. The time aggregated graph that represents this dynamic network is shown in Figure 2(d). In this figure, edge N2-N1 has two attributes, each being a series. The attribute $(1, 2)$ represents the time instants at which the edge is present and $[1, 1, -]$ is the weight time series, indicating the weights at various instants of time. Though this model can include spatial properties at nodes and edges, these properties are not incorporated in the algorithms presented in this paper. Figure 3(a) shows the time aggregated graph (corresponding to Figure 2(a),(b),(c)) and a time expanded graph that represent the same

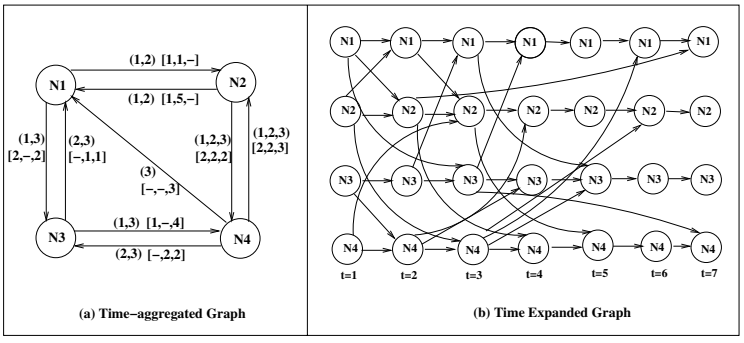

Fig. 3. Time-aggregated Graph vs. Time Expanded Graph

scenario. Edge weights in a time expanded graph are not explicitly shown as edge attributes; instead they are represented by edges that connect the copies of the nodes at various time instants. For example, the weight 1 of edge N2-N1 at $t = 1$ is represented by connecting the copy of node N2 at $t = 1$ to the copy of node N1 at time $t = 2$. The time expansion for the example network needs to go through 7 steps since the latest edge traversal in the network ends at $t = 7$. The traversal 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. If the value of T is very large in a spatial network, it would result in enormously large time expanded networks and consequently slow computations.

Comparison of Storage Costs with Time Expanded Networks: According to the analysis in [20], the memory requirement for a time expanded network is $O(nT)+O(n+mT)$, where n is the number of nodes, m is the number of edges in the original graph, and T is the length of the travel time series. The memory requirement for the time-aggregated graphs would be $O(n + m)T$, assuming an adjacency list representation of the graph. Each edge has a travel time series associated with it, instead of a scalar cost as in the case of a static graph.

This comparison shows that the memory usage of time-aggregated graphs is less than that of time expanded graphs by a factor of $O(nT)$.

2.2 Shortest Path Computation for Time Aggregated Graphs (SP-TAG Algorithm)

In time dependent networks, the shortest path and its traversal time are dependent on the start time at the source node. Here we give an outline of the algorithm that computes the shortest path for a given start time in a time-dependent network. The algorithm uses the time aggregated graph to represent the network. The application of a greedy strategy in the shortest path computation (which is a popular choice in most optimization problems) in a time-aggregated graph

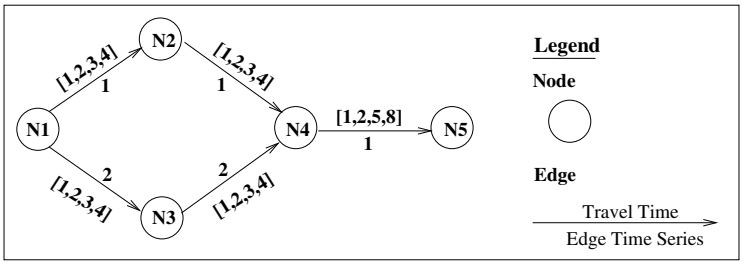

Fig. 4. Optimal Sub-structure of Shortest Paths

faces a challenge. Not all shortest paths display the optimal sub-structure, as illustrated by Figure 4. For the sake of simplicity, the travel times are constant in this example. It can be seen that a shortest path (N1-N3-N4-N5) from N1 to N5 for the start time $t = 1$, which takes 5 time units, does not display optimal substructure. The path from N1 to N4 following the above path is not optimal (shortest path being N1-N2-N4). Although 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 [8]. This result enables us to use a greedy approach to compute the shortest path. The algorithm, called the SP-TAG algorithm, uses greedy strategy to find the shortest path for a fixed start time. Every node has a cost associated with it which represents the travel time to reach the node from the source node. The algorithm picks the node with the least cost and updates the costs of its adjacent nodes. While finding the adjacent nodes, each edge is selected at its earliest available time instant (min_t operation in the algorithm description). A trace of the algorithm is given in Table 1. The table entries are the costs associated with each node (representing the arrival times at the node) at each iteration. The node marked as "closed" is the node with the minimum cost selected for expansion. The travel times are assumed to follow the FIFO property.

Lemma 1: The SP-TAG algorithm is correct.

Proof: As Figure 4 illustrates, the shortest path fails to have optimal structure 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 a 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 2: The time complexity of the SP-TAG algorithm is $O(m(\log T + \log n))$ where T is the number of time instants, n is the number of nodes and m is the number of edges in the time aggregated graph.

Algorithm 1. Shortest Path (SP-TAG) Algorithm

Input:
1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E;
Each node $n \in N$ has a property:
Node Presence Time Series : series of positive integers;
Each edge $e \in E$ has two properties:
Edge Presence Time Series,
Travel_time series : series of positive integers;
$\sigma_{u,v}(t)$ - travel time of edge uv at time t.
2) s: Source node, $s \subseteq N$; 3) d: Destination node, $d \subseteq N$;
4) t_{start}: Start Time;
Output: Shortest Route from s to d for t_{start}
Method:

```
c[s] = t_start; ∀v ≠ s, c[v] = ∞;
// c[u] is the cost at the node u.
Insert s in priority queue Q.
while Q is not empty do {
    u = extract_min(Q);
    for each node v adjacent to u do {
        t = min_t((u, v), c[u]);
        if t + σ_{u,v}(t) < c[v] {
            c[v] = t + σ_{u,v}(t); parent[v] = u;
            if v is not in Q, insert v in Q;
        }
        update Q;
    }
}
}
Output the route from s to d.
```

Proof: 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 $O(\log T + \log n)$. The asymptotic complexity of the algorithm would be

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

The time complexity of the SP-TAG shortest path algorithm based on a time expanded network is $O(nT \log T + mT)$ [3]. It can be seen that the algorithm based on a time-aggregated graph is faster if $\log n < T \log T$.

Table 1. Trace of the SP-TAG Algorithm for the Network shown in Figure 4

Iteration	N1	N2	N3	N4	N5
1	1 (closed)	∞	∞	∞	∞
2	1	2 (closed)	3	∞	∞
3	1	2	3 (closed)	3	∞
4	1	2	3	3 (closed)	6
5	1	2	3	3	6 (closed)

3 Case Study: Best Start Time Shortest Paths

The time dependency of network parameters affects the connectivity and the shortest paths between nodes in a spatial network. As illustrated in Figure 5, the travel time from node N1 to node N3 changes with the start time. If the travel starts at $t = 1$, the commute time would be 6 units. A journey that starts at $t = 1$ reaches N2 at $t = 2$ and waits at N2 until edge N2-N3 becomes available at $t = 5$, thus taking a total travel time of 6 units to reach node N3. The travel on the same route would take 4 units if the start time is moved to $t = 4$. This shows that the shortest paths in a time-dependent network vary with time, which adds an interesting dimension to shortest path computation. A path that takes the smallest travel time for a source-destination traversal over the entire time horizon (called 'Best Start Time shortest Path') can be computed. This is significant since it suggests that it is possible to reduce the travel time for the same source-destination pair if the travel starts at the "right" time instant.

The formulation of algorithms to compute the paths that take the least commute time becomes non-trivial since most of the techniques that are used in static networks might not be applicable in dynamic scenarios. Since the network changes in its parameter values and the topology, meeting the requirements of efficiency and correctness can pose challenges. The potential waits at intermediate nodes can increase the total journey time even if an initial part of the path turns out to be optimal. Figure 5 shows a spatial network that changes with time. The figure shows the snapshots of the network at various instants of time, and the edges are marked with the travel times. It is significant to note that the prefix journeys of the best start time shortest path journey are not always optimal since some optimal prefix journeys can lead to longer waits at intermediate nodes. The best start time for a journey from node N1 to Node N3 is $t = 4$, which takes 4 time units. The optimal path from N1 to N3 that starts at $t = 4$

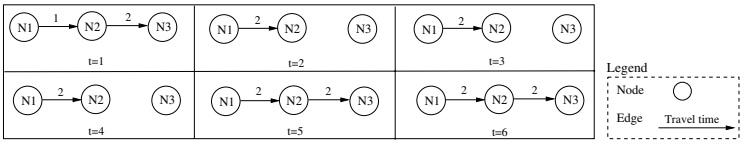

Fig. 5. Network at various instants

is not optimal for the intermediate node N2. The best start time for a path from N1 to N2 is $t = 1$, which proves to be sub-optimal for a journey from N1 to N3. The lack of an optimal substructure in the best start time shortest paths rules out the possibility of using a greedy strategy in the algorithm design.

We propose an algorithm that computes the best start time based on a node-cost time series. The proposed algorithm uses the time aggregated network model to represent a time dependent spatial network.

3.1 BEst Start Time Shortest Path (BEST) Algorithm

While computing the best start time, each node needs to keep track of the travel times to the destination for every start time instant. The proposed algorithm attributes each node with a time series, with i^{th} entry representing the current, least travel time to the destination node for the start time t_i. Due to the lack of optimality of prefix paths and lack of ordering of nodes based on the costs (ie. travel times), nodes cannot be selected and "closed" based on a minimum scalar cost. The algorithm uses an iterative, label correcting approach [1] and each entry in a node time series is modified according to the following condition.

$$C_u[t] = minimum\{C_u[t], \sigma_{uv}(t) + C_v[t + \sigma_{uv}(t)]\} \text{ where, } uv \in E \qquad (1)$$

$C_u[t]$ - Travel time from $u \in N$ to the destination for the start time t.
$\sigma_{uv}(t)$ - Travel time of the edge uv at time t.

The algorithm maintains a list of all nodes that change its cost according to the condition and terminates when there is no further improvement indicated by an empty list. Though the list can be implemented using several data structures, studies on static networks [25,1] have shown that the Two_Q implementation [16] of label correcting algorithms performs the best on road networks.

The search starts at the destination node and proceeds to update the remaining nodes, finally finding the best start time shortest paths from all nodes to the destination. Figure 6 illustrates the trace of the algorithm on a small network. In this example, the destination node is the node N4. The node cost series C_4 is initalized to $[0, 0, 0, 0, 0]$ and the cost series $C_i, i = 1, 2, 3$ are initialized to $[\infty, \infty, \infty, \infty, \infty]$. The nodes that have N4 in their adjacency lists (that is, all nodes N_i such that $N_iN4 \in E$), N2 and N3 are relaxed according to condition (1). These nodes are added to the queue since there is a change in their cost series. The steps continue until the queue is empty, indicating that there is no further cost improvement at any of the nodes. At every iteration, the node that contributes to a cost improvement is stored in a descendant array to facilitate the trace of the shortest paths when the algorithm terminates. At the termination, the cost time series has the travel times for every start time $t = 1, 2 \cdots T$. For example, the cost time series of node N1 shows that the travel times from N1 to N4 for start times $t = 1$ is 4 time units, while the best start time at this node is $t = 4$, which results in a travel time of 2 time units and a best start time shortest path N1-N2-N4. N1-N2 takes 1 time unit at $t = 4$, reaches N2 at $t = 5$ and continues on N2-N4 at $t = 5$, reaching N4 at $t = 6$, taking a total travel time of 2 time units. A more detailed trace is shown in Table 2.

Algorithm 2. BEST Algorithm

Input:

$G(N, E)$: a graph G with a set of nodes N and a set of edges E;
Each node $n \in N$ has a property:
Node Presence Time Series : series of positive integers;
Each edge $e \in E$ has two properties:
Edge Presence Time Series,
Travel_time series : series of positive integers;
$\sigma_{u,v}(t)$ - travel time of edge uv at time t.

Output:

```
Best Start Time shortest route from s to d;
Intialize;
While Queue not Empty
    v = Dequeue();
    For every node u such that uv ∈ E
       For every entry in the cost series C_u of u
          if C_u(t) > σ_uv(t) + C_v(t + σ_uv(t))
             Update C_u(t);
             Enqueue(u);
             Update the descendant array of u.
Find the minimum entry in the node time series.
Return the BestStartTime and the ShortestRoute;
```

Table 2. Trace of the BEST Algorithm for the Network shown in Figure 6

Iteration	N1	N2	N3	N4	Queue
1	$\infty \cdots \infty$	$\infty \cdots \infty$	$\infty \cdots \infty$	$[0, 0, 0, 0, 0]$	$N1$
2	$\infty \cdots \infty$	$[1, 1, 2, 2, 1]$	$[4, 4, 2, 4, 3]$	$[0, 0, 0, 0, 0]$	$N2, N3$
3	$\infty \cdots \infty$	$[1, 1, 2, 2, 1]$	$[2, 3, 2, 4, 3]$	$[0, 0, 0, 0, 0]$	$N3$
4	$[4, 3, 3, 2, 3]$	$[1, 1, 2, 2, 1]$	$[2, 3, 2, 4, 3]$	$[0, 0, 0, 0, 0]$	$N1$
5	$[4, 3, 3, 2, 3]$	$[1, 1, 2, 2, 1]$	$[2, 3, 2, 4, 3]$	$[0, 0, 0, 0, 0]$	–

Lemma 3: The algorithm terminates and computes the best start time paths from every node to the destination.

Proof: The algorithm terminates because there is a positive minimum for the travel time over every path, for every pair of nodes in the network since the edge weights (travel times) are positive and each such path has a finite number of edges. The updates on the costs according to condition(1) will generate the optimal travel times from a node to the destination at the termination of the algorithm. This can be proved by induction on the number of edges on the path. The base condition would be for paths with two edges, say from any node u to the destination node d. Every path with two edges from u to d will transit to some node v and then traverse the edge to d which takes the least time. If we assume the inductive hypotheses is true for every path with k edges, the

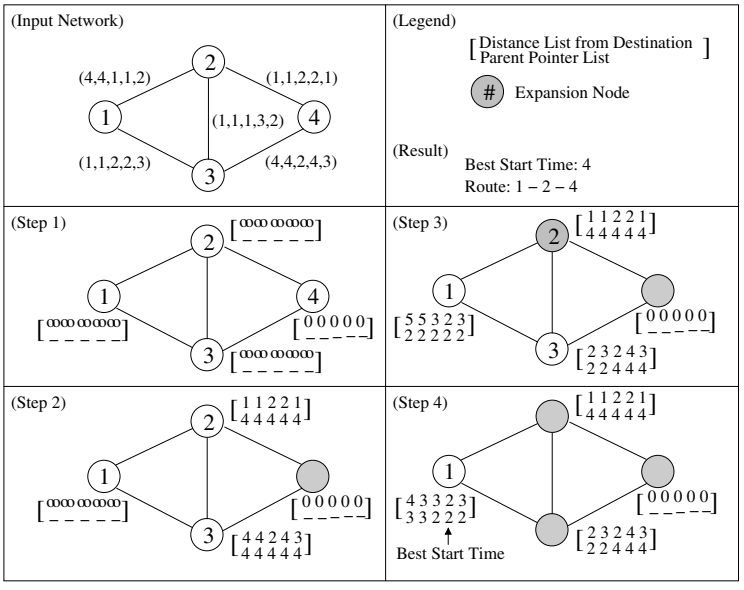

Fig. 6. Trace of the BEST Algorithm

minimality must hold for a path from u with $(k+1)$ edges since we can reach node u that with a minimal $k-$edge path and append uv with travel time $\sigma_{uv}(t)$.

Lemma 4: The computational complexity of the BEST algorithm is $O(n^2mT)$, where n is the number of nodes, m is the number of edges and T is the length of the time series.

Proof: The worst case computational complexity of the label correcting algorithm based on Two-Q data structure is $O(n^2m)$ when the node costs and edge weights are scalar quantities [1]. In the BEST algorithm, the relaxation step operates on a time series (node cost and edge weight) of length T. Hence the computational complexity of the algorithm is $O(n^2mT)$.

4 Experimental Analysis

In this section, the experimental analysis of the BEST algorithm and the SP-TAG algorithm are provided. The purpose of the performance evaluation of the algorithm is to compare the run-times with algorithms based on a time-expanded graph.

Experiment Design. Figure 7 illustrates the experiment design to compare the performance of the proposed algorithm and the algorithm based on a time expanded network. Time expanded graphs make copies of the original network

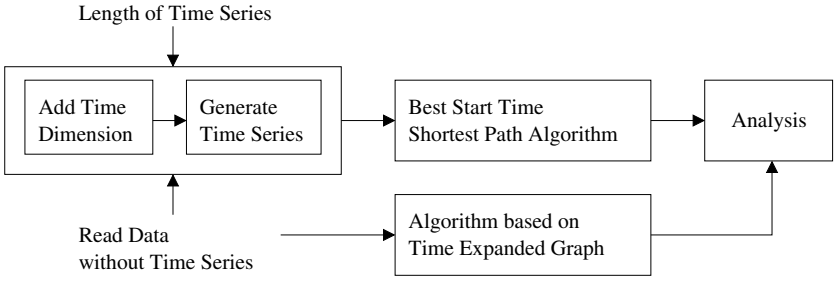

Fig. 7. Experiment Design

for every time instant under consideration. The model used for the proposed algorithm is time-aggregated graphs. In our experiments the following were selected as the independent parameters: 1) network size represented by number of nodes; and 2) the length of the time interval in terms of number of time instants. The data sets have two main components: (1) the network data that consists of the graph structure and (2) the travel time series. The networks chosen are road maps from the Minneapolis downtown area with radii of .5 mile, 1 mile, 2 miles and 3miles. This is appended with travel time series of various lengths. The travel time series were synthetically generated. This data was fed to both a time expanded graph generator, which generates the expanded graph encoding the travel time information. An algorithm for for computing the shortest path for a given start time was run on this graph. The SP-TAG algorithm was run on the same dataset and the results were compared. The time expanded graph was then used to find the start time that results in the least travel time and the results were compared to the results from the BEST algorithm.

The experiments were conducted on a SUN Solaris workstation with 1.77GHz CPU, 1GB RAM and UNIX operating system. Each experimental result reported in the following sections is the average over 5 experiment runs with networks generated using the same input parameters, but with different destination nodes.

4.1 Experimental Results and Anlaysis

We wanted to answer three questions: (1) How does the network size (number of nodes, number of edges) affect the performance of the algorithms? (2) How does the length of the time series affect the performance of the algorithms? (3) How do the the two representations, time expanded graph and time aggregated graph, compare with respect to algorithm performance?

Experiment 1: How does the network size affect the performance of the algorithms?
The purpose of the first experiment was to evaluate how the network size in terms of the number of nodes affects the performance of the algorithms. We fixed the length of the travel time series, and varied the network size to observe

Table 3. Description of Datasets

Dataset	**Radius**	No: of Nodes	**No: of Edges**
1	0.5 mile	111	287
2	1 mile	277	674
3	2 miles	562	1443
4	3 miles	786	2106

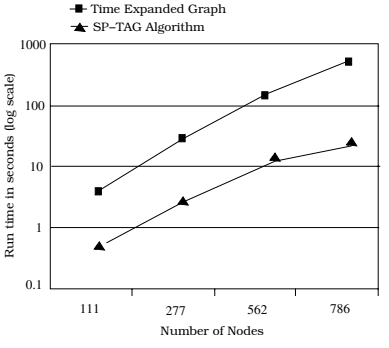

Fig. 8. SP-TAG Algorithm: Run-time With Respect to Network Size

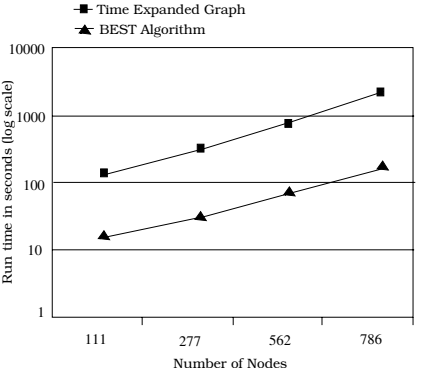

Fig. 9. BEST Algorithm: Run-time With Respect to Network Size

the run times of both the fixed start time(SP-TAG) and best start time(BEST) algorithms and time-expanded graph based algorithms.

The experiment was done with four datasets that represent the road maps from the Minneapolis downtown area of .5 mile, 1 mile, 2 mile and 3mile radius. The length of the time series was fixed at 240. The number of nodes and edges in these datasets are provided in Table 3. Figure 8 shows the run-time of the fixed start time algorithm based on the time aggregated graph and the performance of the algorithm based on the time expanded graph. The SP-TAG algorithm runs faster than the time-expanded graph based algorithm in all cases; further, its run-time seens to increase at a slower rate. Figure 9 shows the performance of the BEST algorithm and that of the time expanded graph algorithm. The run time of the BEST algorithm is much lower than that of the time expanded graph algorithm.

Experiment 2: How does the length of the time series affect the performnace of the algorithms?

In the second experiment, we evaluated how the number of time instants affects the performance of the algorithms. We fixed the network size, and varied the length of the time series to observe the run-time. The number of time instants was varied from 120 to 480 and the network size parameters were fixed at 562 nodes and 1443 edges. As seen in Figure 10, the SP-TAG algorithm performs

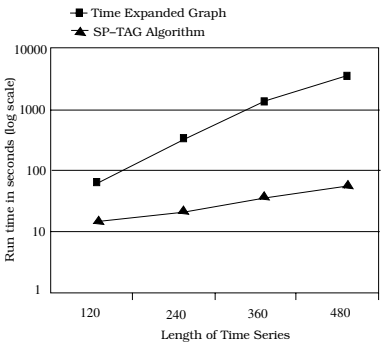

Fig. 10. SP-TAG Algorithm: Run-time With Respect to Length of Time series

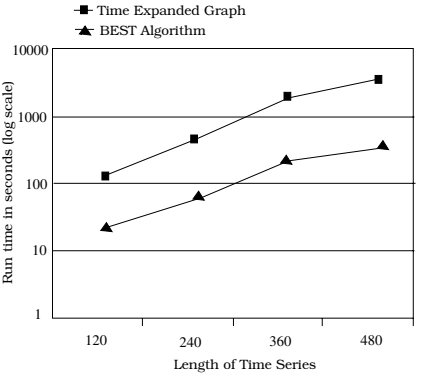

Fig. 11. BEST Algorithm: Run-time With Respect to Length of Time series

better. Figure 11 shows the performance of the BEST algorithm and that of the time expanded graph algorithm. As the length of the time series increases, the number of copies of the entire network required in the case of the time expanded graph increases, resulting in a considerable increase in the size of the entire network, leading to almost exponential increases in run time.

3: How do the the two representations, time expanded graph and time aggregated graph, compare with respect to algorithm performance?
Based on the results of Experiments (1) and (2), it can be seen that algorithms based on the time aggregated graph perform better than those based on the time expanded graph.

5 Conclusions and Future Work

Spatio-temporal networks form a key part of critical applications such as emergency planning and there is a great need for database support in this area. The paper describes a model to represent a spatio-temporal network and proposes two algorithms for shortest path computations. The formulation of these algorithms is based on a model for spatio-temporal networks called time-aggregated graphs. In addition to the algorithm that computes the shortest path for a given start time, we also addressed the time-dependence of shortest paths in networks by formulating an algorithm that computes shortest paths which result in the least travel time over the entire time period. We also present an experimental analysis of the best start time (BEST) algorithm and the fixed start time algorithm (SP-TAG) (which was proposed in [8]). Experiments show that the algorithms based on time aggregated graphs significantly reduce the computational cost compared to similar algorithms based on time expanded networks.

We plan to evaluate the performance of the algorithms using real-traffic datasets shortly. We recently acquired a dataset for interstate highway I-66.

This data contains time-stamped occupancy, speed and volume collected from a number of stations on I-66 on November 6, 2006 using the Advanced Interactive Traffic Visualization System [23]. We anticipate that this evaluation will give new insights into the average case run time of the algorithms, which we expect to be significantly better than the worst case complexity, especially in the case of the BEST algorithm based on a label correcting approach. We are also planning to extend our experiments with Google traffic data and traffic archive data collected by the Traffic Management Center at the University of Minnesota.

The time aggregated graphs can accomodate the time-varying capacities of the road networks. The proposed algorithms need to be extended to give optimal solutions subject to the constraints of time-varying capacities. This would extend the use of the algorithms to domains such as evacuation planning in emergency management, where capacity constraints in the network pose significant challenges. We plan to include spatial attributes at nodes and edges and incorporate necessary changes in the algorithms. We plan to incorporate the algorithms as building blocks that find the shortest paths in the CCRP evacuation planner [13]. We will also explore other graph problems in the context of time aggregated graphs.

Acknowledgments

We are particularly grateful to the members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and valuable suggestions. 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 and Minnesota Department of Transportation. The content does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred.

References

1. Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest Paths Algorithms: Theory and Experimental Evaluation . Mathematical Programming 73, 129–174 (1996)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms (Chapter 26, Flow Networks). MIT Press, Cambridge, MA, USA (2002)
3. Dean, B.C.: Algorithms for minimum-cost paths in time-dependent networks. networks 44(1), 41–46 (2004)
4. Ding, Z., Guting, R.H.: Modeling temporally variable transportation networks. Proc. 16th Intl. Conf. on Database Systems for Advanced Applications , 154–168 (2004)
5. Erwig, M.: Graphs in Spatial Databases. PhD thesis, Fern Universität Hagen (1994)
6. Erwig, M., Guting, R.H.: Explicit graphs in a functional model for spatial databases. IEEE Transactions on Knowledge and Data Engineering 6(5), 787–804 (1994)
7. ESRI. ArcGIS Network Analyst (2006), `http://www.esri.com/software/arcgis/extensions/`

8. George, B., Shekhar, S.: Time-aggregated Graphs for Modeling Spatio-Temporal Networks - An Extended Abst ract . Proceedings of Workshops at International Conference on Conceptual Modeling (2006)
9. Hamre, T.: Development of Semantic Spatio-temporal Data Models for Integration of Remote Sensing and in situ Data in Marine Information System. PhD thesis, University of Bergen, Norway (1995)
10. Kaufman, D.E., Smith, R.L.: Fastest paths in time-dependent networks for intelligent vehicle highway systems applications. IVHS Journal 1(1), 1–11 (1993)
11. Kohler, E., Langtau, K., Skutella, M.: Time-expanded graphs for flow-dependent transit times. In: Proc. 10th Annual European Symposium on Algorithms, pp. 599–611 (2002)
12. Sellis, T., Koubarakis, M., Frank, A., Grumbach, S., Güting, R.H., Jensen, C., Lorentzos, N.A., Manolopoulos, Y., Nardelli, E., Pernici, B., Theodoulidis, B., Tryfona, N., Schek, H.-J., Scholl, M.O.: Spatio-Temporal Databases: The CHOROCHRONOS Approach. In: Sellis, T., Koubarakis, M., Frank, A., Grumbach, S., Güting, R.H., Jensen, C., Lorentzos, N.A., Manolopoulos, Y., Nardelli, E., Pernici, B., Theodoulidis, B., Tryfona, N., Schek, H.-J., Scholl, M.O. (eds.) Spatio-Temporal Databases. LNCS, vol. 2520, Springer, Heidelberg (2003)
13. Lu, Q., George, B., Shekhar, S.: Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Resu lts. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, Springer, Heidelberg (2005)
14. Oracle. Oracle Spatial 10g, An Oracle White Paper. August (2005), http://www.oracle.com/technology/products/spatial/
15. Orda, A., Rom, R.: Minimum weight paths in time-dependent networks. networks 21, 295–319 (1991)
16. Pallottino, S.: Shortest-Path Methods: Complexity, Interrelations and New Propositions. Networks 14, 257–267 (1984)
17. Pallottino, S., Scuttella, M.G.: Shortest path algorithms in tranportation models: Classical and innovative aspects. Equilibrium and Advanced transportation Modelling , 245–281 (1998)
18. Rasinmäki, J.: Modelling spatio-temporal environmental data. In: 5th AGILE Conference on Geographic Information Science, Palma, Balearic Islands, Spain (April 2002)
19. Shekhar, S., Chawla, S.: Spatial Databases: Tour. Prentice-Hall, Englewood Cliffs (2003)
20. Sawitzki, D.: Implicit Maximization of Flows over Time. Technical report, University of Dortmund (2004)
21. Dreyfus, S.E.: An appraisal of some shortest path algorithms. Operations Research 17, 395–412 (1969)
22. Shekhar, S., Liu, D.: CCAM: A Connectivity-Clustered Access Method for Networks and Networks Computations. IEEE Transactions on Knowledge and Data Engineering, 9 (January 1997)
23. Spatial Data Management Lab, Virginia Polytechnic Institute and State University. AITVS: Advanced Interactive Traffic Visualization System (2007), http://spatial.nvc.cs.vt.edu/traffic_zhh/
24. Stephens, S., Rung, J., Lopez, X.: Graph data representation in oracle databese 10g: Case studies in life sciences. IEEE Data Engineering Bulletin 27(4), 61–66 (2004)
25. Zhan, F.B., Noon, C.E.: Shortest Paths Algorithms: An Evaluation Using Real Road Networks. Transportation Science 32, 65–73 (1998)