

Chapter 2

Time Aggregated Graph: A Model for Spatio-temporal Networks

Abstract Spatio-temporal networks represent networks where entities have spatial attributes and the topology and parameters display time-dependence. Given the significance of such networks in critical domains such as transportation science and sensor data analysis, the importance of a model that is simple, expressive and storage efficient to represent such networks cannot be understated. The model must provide support for the design of algorithms to process frequent queries that need to be answered in the application domains. 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. 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. The chapter presents a description and comparison of these models.

2.1 Modeling Spatio-temporal Networks

The growing importance of application domains such as transportation networks, emergency planning and location based services 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 efficient algorithms that implement the frequent operations posed on such networks. Frequent queries on such networks might include finding 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 a 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 accommodate 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 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 s. 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.

Various temporally enhanced entity relationship models have been proposed [19]. Some of these models capture the temporal properties of relationships in terms of their existence and validity periods; these do not explicitly capture the changes in relationship types. Other models such as TERC+ [50] capture the temporal nature of relationship types by expressing the relationship changes in terms of entity transformations. This model basically uses entity subtypes to represent temporal evolution of entities as well as relationships and hence might not be able to represent evolving relationships between entities without subtypes.

In this chapter a spatio-temporal network model named time aggregated graph [16, 17] is described. 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. The representational capability of the model is illustrated through various application domains such as transportation science and emergency planning. The model is compared with another graph-based model, the time expanded graph, in the context of various application domains.

2.1.1 Illustrative Application Domains

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

Table 2.1 Example Queries with Time-variance and Flow Networks

	Static	Time-variant
Graph (no capacity constraints)	Which is the shortest travel time path from Minneapolis downtown to airport?	Which is the shortest travel time path from Minneapolis downtown to airport at different times of a work day?
Flow network	What is the capacity of Twin-Cities freeway network to evacuate Minneapolis downtown?	What is the capacity of Twin-Cities freeway network to evacuate Minneapolis downtown at different times in a work day?

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. There are many factors affecting the start time and the shortest route such as congestion levels, incident location, and construction zone. This is illustrated by the simple time-variant network shown in Fig. 2.2. It can be seen that the travel time from node N1 to node N2 changes with the start time. If the travel starts at $t = 1$, the commute time would be 6 units; travel on the same route would take 4 units if the start time is moved to $t = 3$. This shows that the shortest paths in a time-dependent network vary with time which adds a new dimension to shortest path computation which cannot be ignored. 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 into the models and algorithms related to transportation networks. One of the greatest challenges in transportation science is how to manage traffic in time-varying transportation networks, especially in disaster situations. Popular models of emergency traffic use time-variant flow-network [1] operations like min-cut and max-flow [5]. The queries typically encountered in emergency traffic management would involve time-variant properties, as illustrated in Table 2.1.

In crime analysis and prevention, identifying the areas of increasing criminal activity is a key step. Computing the routes that show significantly high crime rates can improve the efficiency of the patrol operations. Crime data usually consists of the geographical location of the crime, type of crime and its time of occurrence [28]. To compute the routes of high criminal activity, a model is required to represent the underlying transportation network along with the time dependent crime data associated with its edges and nodes. For example, the crime rates can vary with the time of the day and the interesting routes can change. With the availability of time-varying data, it becomes important to incorporate this data in the models and analysis of crime data.

Another interesting area of exploration is the effect of temporal dimension on conceptual models such as Entity-Relationship (ER) model [3] and more specifically

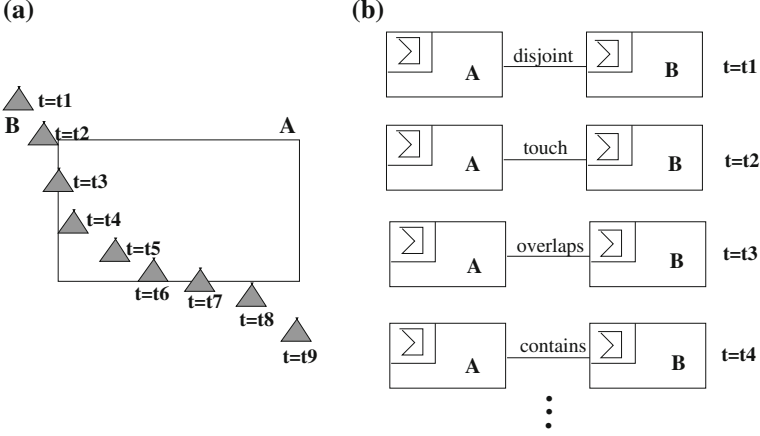


Fig. 2.1 Illustration of a dynamic relationship between two objects and its representation. **a** Location of Moving Object. **b** PEER Diagrams for (a)

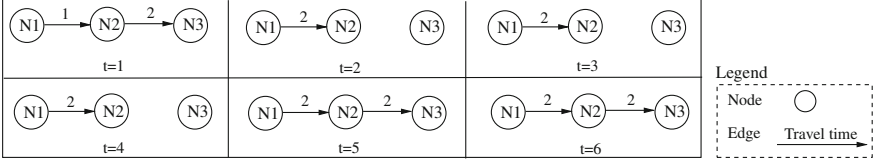


Fig. 2.2 Network at various instants

on the Pictogram-Enhanced Entity-Relationship (PEER) diagram [46]. A simple example is shown in Fig. 2.1. It illustrates a scenario where a moving sensor B crosses a geographic area A. Figure 2.1a shows the locations of B at discrete time instants ($t = t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9$). The relationship of object B with object A changes with time. This has been represented in Fig. 2.1b using a series of PEER diagrams. Each diagram represents the relationship at an instant. For example, the first diagram represents the time instant $t = t_1$ when the relationship between the objects is ‘disjoint’. The figure shows the representations for the first four instants; the rest are modeled in a similar manner.

2.1.2 Broad Computer Science 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 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. There are also be cases where the road segments are unavailable periodically due to traffic management strategies such as using all lanes of a street in the same direction to handle peak time congestion. Conventional graph algorithms cannot easily be applied to the snapshots at discrete time instants to evaluate frequent queries without accounting for relationships among snapshots.

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 prefix property (a requirement for the greedy approaches [5]) 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 the potential wait at the intermediate node. In the network shown in Fig. 2.2, the best time to start 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 $t = 4$ 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 optimal prefix property in best start time shortest paths rules out the possibility of using a greedy strategy in algorithm design.

Key Features Of TAG:

Graph Aggregation: The temporal variation in the topology and parameter values can be represented using aggregates as edge/node attributes in the graph used to represent the spatial 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.

Query Language: A query language needs to represent common queries. A key challenge is to define a complete set of logical operators for the time-aggregated graph.

Query Processing: The time aggregated graph with the proposed query operators will be used to process queries pertaining to the domain applications. A frequent query that arises in spatio-temporal networks is the shortest path computation. The algorithm needs to consider the availability of the required edges and nodes at the appropriate time instants. If the shortest route and the shortest route travel time are time-dependent, shortest path computation can be performed for a given start or it can find the least travel time path over the entire time period of interest.

In this chapter we describe a model for spatio-temporal networks called the time aggregated graph based on graph aggregation. The time-aggregated graph keeps track of the time-dependence of a graph through a time series attached to each node and edge that indicates their presence at various instants of time. 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 define a set of logical operators based on the time aggregated graph.

2.2 Basic Concepts

Spatial networks that show time-dependence serve as the underlying networks for most location based services. Traditionally graphs have been extensively used to model spatial networks (e.g. road networks) [40]; 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. Formulation of computationally efficient and correct algorithms for the shortest path computation that takes into account the dynamic nature of the networks is important. 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 the set of frequent queries posed by an application on a spatial network and the patterns 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 spatial networks called “time aggregated networks” [16]. The algorithm presented in this paper is 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.

2.2.1 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 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.

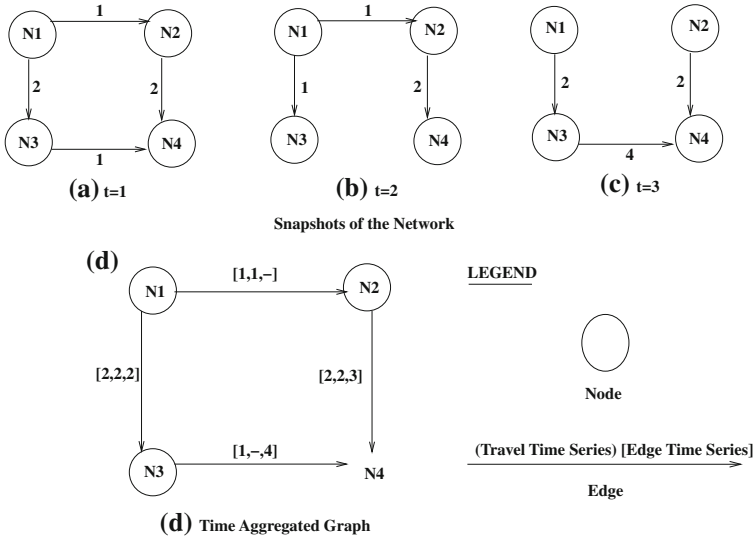


Fig. 2.3 Network at various time instants and the time aggregated graph

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

$g_1 \dots g_l$ are mappings from edges to the time series associated with the edges, and

$w_1 \dots w_p$ indicate the time dependent weights (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. It is assumed 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 2.3a–c shows a network at three time instants. The network topology and parameters change over time. For example, edge N3–N4 is present at time instants $t = 1, 3$, and disappears at $t = 2$ and its weight changes from 1 at $t = 1$ to 4 at $t = 3$. The time aggregated graph that represents this dynamic network is shown in Fig. 2.3d. In this figure, edge N3–N4 has two attributes, both being a series. The attribute (1, 3) represents the time instants at which the edge is present and $[1, \infty, 4]$ is the weight time series, indicating the weights at various instants of time. Figure 2.4a shows the time aggregated graph (corresponding to Fig. 2.3a–c and the time expanded graph that represent the same 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

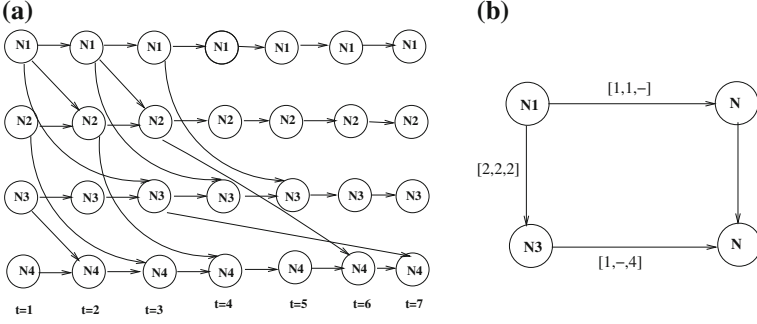


Fig. 2.4 Time-aggregated graph versus time expanded graph. **a** Time Expanded Graph. **b** Time-aggregated Graph

1 of edge N1–N2 at $t = 1$ is represented by connecting the copy of node N1 at $t = 1$ to the copy of node N2 at time $t = 2$. 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 of 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.

2.2.2 A Logical Data Model

Basic Graph Operations

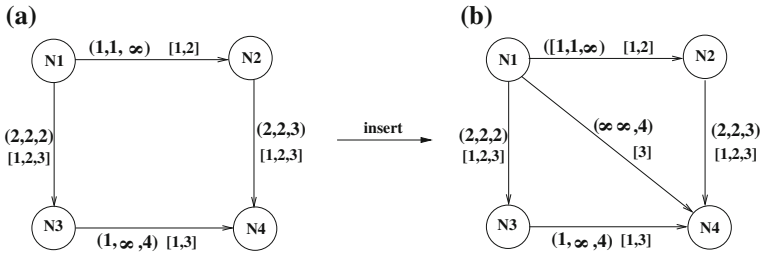
We extend the logical data model described in [40] to incorporate the time dependence of the graph model. The framework of the model consists of two dimensions (1) graph elements, namely node, edge, route and graph and (2) operator categories that consist of accessors, modifiers and predicates. A representative set of operators for each operator category is provided in Tables 2.2, 2.3 and 2.4. Table 2.2 lists a representative set of ‘access’ operators. For example, the operator *getEdge*(*node1*,*node2*,*time*) returns the edge properties of the edge from node *node1* to node *node2*, such as the edge identifier (if any) and associated parameters at the specified time instant. For example operator *getEdge*(N1,N2,1) on the time-aggregated graph shown in Fig. 2.3 would return the travel time of the edge N1–N2 at $t = 1$, that is 1. Similarly, *get_edge*(*node1*,*node2*) returns the edge properties for the entire time interval. In Fig. 2.3, the operator *get_edge*(N1,N2) would result in (1, 1, ∞). *get_edge_earliest*(N3,N4,2) returns the earliest time instant at which the edge N3–N4 is present after $t = 2$ (that is $t = 3$). Table 2.3 shows a set of modifier

Table 2.2 Examples of operators in the accessor category

	at_time	at_all_time	at_earliest
Node	get(node,time)	get_node(node)	get_node_earliest_Presence (node,time)
Edge	getEdge(node1,node2,time)	get_edge(node1,node2)	get_edge_earliest_Presence (node1,node2,time)
Route	getRoute(node1,node2,time)	get_route(node1,node2)	get_route_earliest_Presence (node1,node2,time)
Graph	get_Graph(time)	get_Graph()	–

Table 2.3 Examples of operators in the modifier category

	Insert		Delete		Modify	
	at_time	at_all_time	at_time	at_all_time	at_time	at_all_time
Node	insert(node, time,value)	insert(node, valueseries)	delete(node, time)	delete(node)	update(node, time,value)	update(node, valueseries)
Edge	insert(node1, node2, time,value)	insert(node1, node2, valueseries)	delete(node1, node2, ,time)	delete(node1, ,node2)	update(node1, node2,time value)	update(edge, valueseries)
Route	insert(node1, node2,time)	insert(node1, ,node2)	delete(node1 ,node2,time)	delete(node1, node2)	–	–
Graph	insert(graph time)	insert(graph)	delete(graph, time)	delete(graph)	update(graph, ,time)	update(graph)

**Fig. 2.5** A time aggregated graph before and after an insert operation. **a** Before insert, **b** after insert

operators that can be applied to the time aggregated graphs. For example, Fig. 2.5a, b show a time aggregated graph before and after the *insert(N1,N4,3,4)* operation this operation inserts edge N1–N4 at time instant $t = 3$ and the edge cost is 4. 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 2.4 Predicate operators 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	exists(node u,node v, at_time_t)	exists(node u,node v, after_time_t)
Route	exists(node u,node v,a_route r at_time_t)	exists(node u,node v,a_route r, after_time_t)

Table 2.4 illustrates these operators. 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. *exists(N1,N2,1)* on the time aggregated graph in Fig. 2.3 returns a “true” since the edge N1–N2 exists at $t = 1$.

The fundamental entities in graphs, namely, *Graph*, *Node*, and *Edge* and a series of 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.
}

```

2.2.3 Physical Data Model

A static graph $G = (V, E)$ can be represented using an adjacency matrix. This is a $|V| \times |V|$ matrix, A such that the element a_{ij} is defined as $a_{ij} = w_{ij}$ if $ij \in E$ and w_{ij} is the weight of the edge ij and

$a_{ij} = 0$, otherwise. This representation requires $O(N^2)$ memory. It can be seen that the storage required for this representation is independent of the number of edges in the graph, in relation to the number of nodes. In other words, there is no saving in memory even when the graphs are sparse. One representation that can exploit such sparsity is the adjacency list representation. The adjacency list representation of a graph $G = (V, E)$ consists of an array of lists, one for each vertex $v \in V$. The list corresponding to a vertex v contains all vertices that are adjacent to v in G . For a directed graph, the space requirement for the lists is $O(m)$ where $m = |E|$. The total memory requirement is $O(n + m)$ where $n = |V|$. The weight of each edge uv is stored with the vertex v in u 's adjacency list. This representation is especially suitable for sparse graphs.

Time aggregated graphs can be represented by either one of the representation, with the necessary modifications. These representations need to be extended to include the time series representations on edges (corresponding to time dependent edge costs) and nodes. Adjacency list representation is extended by adding a list to each vertex in the adjacency list. Adjacency list representation uses an array of pointers one pointer for each node. The pointer for each node points to a list of immediate neighbors. Stored at each neighbor node are the edge presence series and travel times for the edge starting from the first node to this neighbor. Since the length of the time series is T , where T is the length of the time period, the adjacency list representation would require $O(m + n + nT + mT)$, where n is the number of nodes and m is the number of edges. In reality, not all time series would be of length T and assuming an average length α , the storage would be $O(n + m + \alpha n + \alpha m)$. The time series store a single value if the value of the attribute remains constant, indicated by the character 'F'. If the value of the attribute changes over time, it is indicated by the character 'V'.

To extend the adjacency matrix to represent the time aggregated graph, a third dimension can be added. The new matrix A would be $n \times n \times T$, requiring $O(n^2T)$ memory. Figure 2.6a, b show the adjacency list and adjacency matrix representations for the time aggregated graph shown in Fig. 2.3. For example, the edge N1–N2 in the graph at $t = 1$ is represented by the pointer from N1 to N2 in the adjacency list. The array $(1, 2, \infty)$ is stored at N2 to represent the travel times at $t = 1, 2, 3$ for the edge N1N2. In the adjacency matrix the presence of edge N1N2 at a time instant $t = 1$ is represented by $A[1, 2, 1] = 1$, since the travel time for the edge is 1 unit at $t = 1$. Since the edge is absent at an instant $t = 3$, $A[1, 2, 3] = \infty$ which indicates an infinite edge cost at time instant $t = 3$. Note that the start node, the end node and the time instant are represented by the first, second and the third dimension of the matrix. Though the adjacency matrix has been illustrated as three snapshots in Fig. 2.6b for the sake of clarity, they are represented in one, three-dimensional matrix.

Logical operations on a time-aggregated graph can be classified as

1. Topology first operators (graph dominated operations).
Examples include `get_route(n1,n2)` and `get_edge(n1,n2)`.
2. Time-first operators (Time dominated queries).

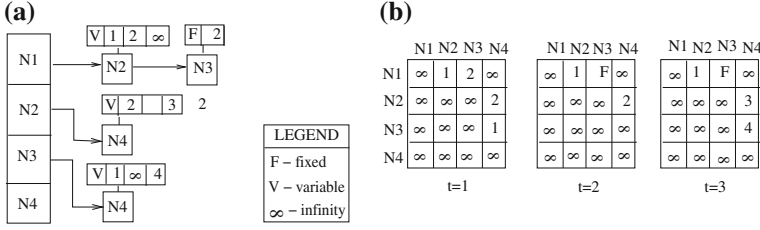


Fig. 2.6 Storage structures for time aggregated graph. **a** Adjacency list representation, **b** adjacency matrix representation

Some examples are `get_Graph(time t)` and `get_edge_at_t(n1,n2,t)`.

Both representations are equally capable of handling graph dominated queries. To compute time first operations (snapshot queries such as to find the graph at a given time instant), adjacency matrix representation is more suitable. In this representation, these queries represent the time slices of the matrix at the given time instants.

Graphs representing transportation networks are generally sparse and hence adjacency list representation is more likely to be storage efficient compared to adjacency matrix representations. The choice is hence a tradeoff between the storage cost and the frequency of time dominated queries. We expect route queries (which are topology first queries) to be more frequent and since adjacency list representation is capable of handling these, based on storage costs, we used adjacency lists in our implementations. Moreover, most databases use adjacency list representation.

2.2.3.1 Towards Handling Infinite Time Series

In most domains that involve spatio-temporal networks such as transportation networks, crime data analysis, and sensor networks data is continuously collected at discrete instants of time. For example, sensors on urban highways measure congestion levels every 30s and crime data is appended with every time a crime occurs.. Conceptually, the time aggregated graph can be viewed as a time series of graphs. Each graph represents the attribute values and the topological structure of the network at the given instant of time. Based on the periodicity of data collection, the application domains can be broadly classified into (1) applications where data is measured periodically and (2) applications such as crime analysis where data is recorded when an event occurs.

When data is measured periodically, the underlying model should be able to capture the changes that take place in the spatio-temporal network at every instant. Time aggregated graphs represent this as a time series of graphs, each graph in the series modeling the state of the network. For example, the state of a road network at $t = t_1$ would be represented as a graph corresponding to this instant. The state of a sensor network, which would include the measurements at an instant would also be modeled in a similar manner.

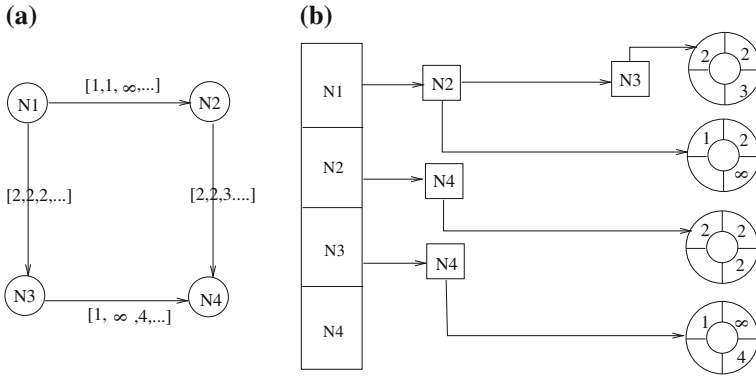


Fig. 2.7 Representation of sliding windows in time aggregated graph. **a** Time aggregated graph, **b** implementation of time series

In application domains where the network state changes due to an event, the time aggregated graph stores the tuples of time stamp and the event.

Implementation

The time series of graphs would be implemented as a graph where the node and edge attributes are time series. Most application domains deal with ‘infinite’ streams of data, and the edge and node attributes are possibly infinite time series. One implementation uses sliding windows implemented through circular buffers. Figure 2.7a shows a time aggregated graph with time series attributes on its edges. Figure 2.7b shows the modified adjacency list representation that implements an infinite time series. Each time series is stored in a circular buffer.

2.3 Evaluation and Validation

2.3.1 Representational Comparison: Time Aggregated Graphs Versus Existing Models

A time-expanded network has one copy of the set of nodes for each discrete time instant. Corresponding to each edge with transit time t in the original network, there is a copy of an edge (called the cross edge) between each pair of copies of nodes separated by the transit time t [14, 22, 26]. Thus, a time-dependent flow in a dynamic network can be interpreted as a static flow in a time expanded network. This allows application of static algorithms on such networks to solve dynamic flow problems. Apart from the “enormous increase in the size of the underlying network” [26] the suitability of the model in some application domains needs further exploration.

A time expanded graph assumes that the edge weight represents a flow parameter, and it represents the time taken by the flow to travel from the source node to the end node. This is represented by the cross edges between the copies of the graph.

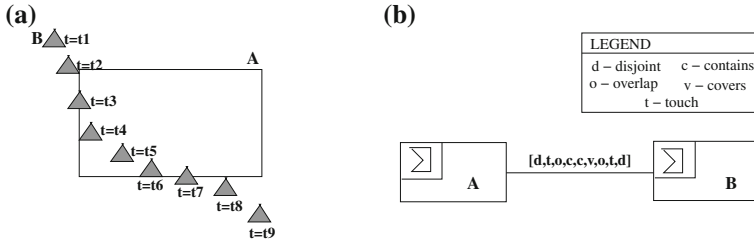


Fig. 2.8 Illustration of a dynamic relationship between two objects and its representation. **a** Locations of Moving Object. **b** TAG Representation for (a)

Since the cross edges in a time expanded graph represent a flow across the nodes, the representation of non-flow networks using this model is not obvious. By contrast, the time aggregated graph model does not impose such a restriction because the attributes are collected into a time series. This difference can be illustrated through the example of the possible extension of the PEER diagram explained in Sect. 2.1.1. While time aggregated graph would model the time-dependent relationships as a time series on the edge connecting the nodes (that represent the entities), the representation of the same scenario is not obvious when time expanded graphs are used. An illustration of the representation of time-dependent relationships using time-aggregated graph representation for the scenario depicted in Fig. 2.1 is shown in Fig. 2.8. Figure 2.1 shows the locations of B at discrete time instants ($t = t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9$). The relationship of object B with object A changes with time. This has been represented in Fig. 2.8a using an aggregated representation. The line segment that represents the relationship has an attribute which is an ordered set, each element indicating the current relationship of object B with A. For example, the second entry 'o' indicates that the object B touches A at $t = t_2$ and overlaps A at $t = t_3$. In the domain of crime analysis, the number of crimes reported on a road segment (represented by an edge) at a given time might not be meaningfully represented by an edge in the time expanded graph. The time aggregated graph would represent this as an element in its time series attribute.

In most spatio-temporal networks, the length of the time period (indicated by T in this paper) might not be known in advance since data arrives as a sequence at discrete time instants. For example, sensors in transportation networks collect data at a rate of about once every 30s. Crimes are reported whenever an incident occurs. In addition to being able to represent these attributes, the model must be capable of handling infinite sequences of data. Since time expanded networks require a prior estimate of the length of the time period T , handling of infinite time series might not be easy and obvious. Also, the necessity for the prior knowledge of T might lead to problems in the algorithms based on time expanded networks since an underestimation of T can result in failure of finding a solution. On the other hand, an over-estimated T will result in an over-expanded network and hence lead to unnecessary storage and run-time and would adversely affect the scalability of the algorithms.

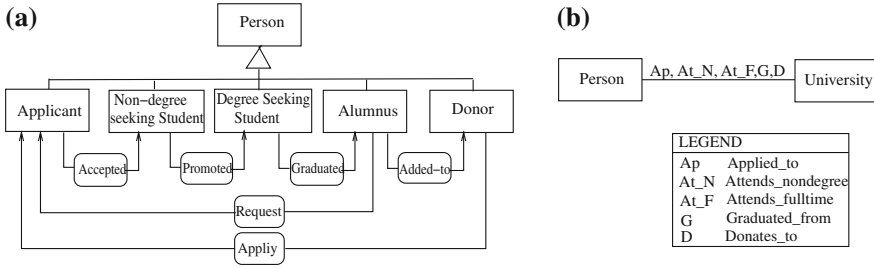


Fig. 2.9 Representations of dynamic relationships in TERC and aggregated graph. **a** An example TERC model, **b** aggregated model (**a** adapted from [50])

Time expanded graphs model the time-dependence of edge parameters through the cross edges that connect the copies of the nodes. This representation, thus, does not provide the means to separate data (for example, an edge attribute series) from its physical representation and hence can adversely affect physical data independence.

The temporal conceptual model TERC+ [50] models dynamic relationships between entities using evolutions of the entities involved. The temporal nature is captured through representing transitions of objects. An example is shown in Fig. 2.9. It represents a dynamic relationship between a person and a University. The relationship changes from an applicant to a donor after graduation. The change in the relationship is represented through various classes of the same entity as shown in Fig. 2.9a. An aggregated model of the same scenario is shown in Fig. 2.9b. Though at the finest level, the representations would be the same, the aggregated model facilitates a better high level summarization. This model might not be sufficient to represent cases where entity subtypes cannot be used to model evolving relationships. For example, Fig. 2.1, represents a scenario where the entities (a sensor and a geographic area) involved in the dynamic relationship do not have subtypes and hence might not yield itself to this model.

2.3.2 Comparison of Storage Costs with Time Expanded Networks

According to the analysis in [41], 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 framework of a time aggregated graph would require a memory of $O(n + m)$, where n is the number of nodes and m is the number of edges. Each edge that has a time-varying attribute has an attribute time series associated with it. If the average length of the time series is $\alpha (\leq T)$, the memory required is $O(\alpha m)$, assuming an adjacency list representation. The total memory requirement for a time aggregated graph is $O(n + m + \alpha m)$. This comparison shows that the memory usage of time-aggregated graphs is less than that of time expanded graphs $nT > n$ and $\alpha \leq T$.

Fig. 2.10 TAG: storage costs

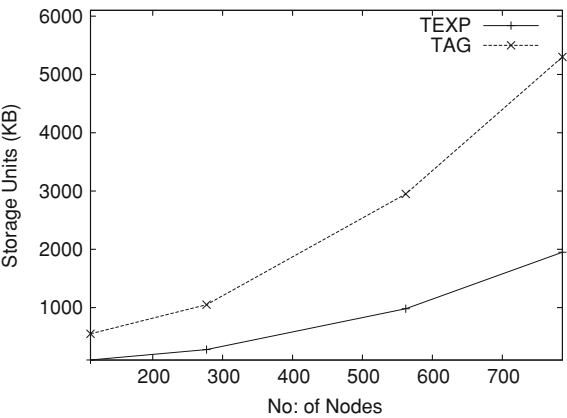


Table 2.5 Description of datasets

Dataset	Radius (miles)	Number of nodes	Number of edges
1	0.5	111	287
2	1	277	674
3	2	562	1443
4	3	786	2106

Experimental Evaluation:

Figure 2.10 shows the result of evaluation of storage requirement for time aggregated graph in comparison with time expanded graph. The networks used are the road maps from the Minneapolis downtown area, of radii, 0.5, 2 and 3 miles. This is appended with travel time series of length 200. The number of nodes and edges in these datasets are provided in Table 2.5.

The evaluation shows that memory cost of TAG is much less than the time expanded graph.

2.4 Summary

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. This chapter describes a model based on time aggregation to represent a spatio-temporal network. Time aggregated graphs represent the time variant properties by aggregating edge and node parameters into time series. The analytical and experimental analysis of storage cost requirements of time aggregated graph are presented.



<http://www.springer.com/978-1-4614-4917-1>

Spatio-temporal Networks

Modeling and Algorithms

George, B.; Sangho, K.

2013, XII, 73 p. 47 illus., Softcover

ISBN: 978-1-4614-4917-1