High Performance Multimodal Networks

Erik G. Hoel, Wee-Liang Heng, and Dale Honeycutt

Environmental Systems Research Institute,
380 New York Street, Redlands, CA 92373
{ehoel, wheng, dhoneycutt}@esri.com

Abstract. Networks often form the core of many users' spatial databases. Networks are used to support the rapid navigation and analysis of linearly connected data such as that found in transportation networks. Common types of analysis performed on such networks include shortest path, traveling salesman, allocation, and distance matrix computation.

Network data models are usually represented as a small collection of tables: a junction table and an edge table. In the context of networks used to model transportation infrastructure, it is also necessary to model turn restrictions and impedances (delays). Network data is frequently persisted in normalized relational tables that are accessible via standard SQL-based queries. We propose a different approach where the network connectivity information is persisted using a compressed binary storage representation in a relational database. The connectivity information is accessible via standard Java, .NET, and COM APIs that are tailored to common access patterns used in the support of high performance network engines. These network engines run on the client or application server tier rather than as extensions on the relational server.

In this paper, we discuss the problem of building a robust and scalable implementation of a network data model. The fundamental and central requirements are enumerated. These requirements include support for hierarchical networks, turn restrictions, and logical z elevations. We propose a different approach to representing network topology that addresses many of the high-end modeling requirements of network systems. Our approach supports all of the listed requirements in addition to multimodal modeling (e.g., coexistent road, bus, and rail networks) within the context of multi-user, long transaction databases.

1 Introduction

Network data models have been used to represent geographic information for well over thirty years [15], [18], [19]. These models have been incorporated into a number of operational systems (see, for example TransCAD [3] or ARC/INFO [22]). Despite the relative maturity of such technology, most systems have fallen short of meeting the most sophisticated requirements of transportation network modeling. Such requirements include the ability to model multimodal (or intermodal) transportation systems (transportation networks where two or more different transportation modes are linked – e.g., roads and rail) and the ability to handle coincident features participating in different modes of the model (e.g, subways underneath streets, or bus

C. Bauzer Medeiros et al. (Eds.): SSTD 2005, LNCS 3633, pp. 308 – 327, 2005.

routes along city streets) when geometric analysis of the participant features is used to derive network connectivity [29]. In addition, some systems fail to address the requirement to model turns and maneuvers without applying complex graph transformations to represent permissible turns as explicit edges [30].

In this paper, we describe a design for modeling multimodal networks that are persisted in relational databases. This design is the basis for our implementation of networks in the ArcGIS geographic information system. The design satisfies the fundamental goal of supporting sophisticated network models that are consumed by high performance network engines, and is tailored for fast retrieval of connectivity information within network analysis algorithms. Network engines provide fine grained (i.e., forward star [6]) access to very large external networks persisted in an RDBMS or the file system, and are intended to reside on the client in the case of traditional two-tier systems, or on the application server in *n*-tier architectures. The network engine supports a rich set of network analysis algorithms, such as shortest path finding, traveling salesman problems, and network resource allocation operations, that also execute in the desktop or application server tier and are used in a variety of desktop and server-based network analysis applications.

In the first section of this paper, we review the logical model of network topology. The major requirements of a high-end network data model are discussed and a connectivity model that supports multimodal networks is presented. We then consider the issue of representing turn restrictions and maneuvers (multi-part turns) – a critical component in transportation networks [9]. Existing approaches to representing turns are reviewed and our modeling approach is presented. The access model of consuming networks is reviewed in the context of common workflows as well as a query model that is tailored to the support of high performance network engines. We then address the issue of the physical storage representation of a network. The conventional physical database implementation is briefly detailed. We then present our alternative physical representation and highlight the reasons and motivation behind its departure from the conventional implementation. We conclude with a brief discussion of our implementation experience and outline our ongoing future research in this domain.

2 Logical Model

The movement of people, the transportation of goods and services, as well as the distribution of resources, energy, and communication are commonly modeled with network systems. Network data structures for representing geographic information are a standard topic in geographic information science [18], [27].

In this paper, we use the term *network* to refer to a connectivity graph of junctions and their connecting edges, where each junction and edge is associated with a feature with point or line geometry respectively. The term *network element* is used to refer to the collection of junctions and edges comprising the network. All network elements have a set of numeric properties, called *network attributes*. Attributes capture information about network elements, such as the travel time across an element, and are used to define the navigation context during an analysis.

Junction attribute values provide a high-level view of traversing intersections. For example, the travel time attribute value on a junction element describes how long it takes to cross the element, ignoring the edge elements used to enter and exit the junction element. For more detailed modeling of traversing intersections, we use turns. In the simplest case, a turn element models entering a junction from a particular edge element and exiting to another. A multipart turn element, also known as a maneuver, enters the junction element from a path of two or more connected edge elements.

Turn elements are not strictly part of the graph model. They represent a relationship rather than being an abstraction of a real-world entity. Turns do not modify the junction-edge connectivity of the network; instead they affect traversability of the network elements. Turns are not considered an attribute of a network junction, though they occur at every junction. This is because they are intrinsically dependent upon the properties of the associated network edges.

The connectivity graph of a network is derived from the source data during a process called *network building*. During a build, junction, edge and turn elements are generated from point, line and turn features, and connectivity relationships are established. The connectivity graph is typically stored separately from the source data, with network analysis algorithms (including the build process) consuming it.

2.1 Requirements

The primary requirements for any robust implementation of network data models are:

- **Multimodal models.** In the context of transportation networks, a multimodal network is one in which two or more types of transportation modes (such as walking, riding a train, or driving a car) are modeled. Alternatively, with utility networks, a multimodal network may consist of the differing transmission and distribution systems.
- **Hierarchical models.** Hierarchy is used within network models to further control the flow within the network [19]. Differing elements may be assigned to different levels of hierarchy, with flow through the higher levels of the hierarchy taking precedence over the lower levels when performing path or route finding operations. Within transportation networks, interstate highways are commonly associated with the highest level of the hierarchy, state highways and major feeders the next lower level, and city streets the lowest level of the hierarchy.
- **Turns and maneuvers.** Support for turning movements, both two-part turns and multi-part turns (known as maneuvers), is necessary in order to more accurately model transportation networks. The definition of a turn should be separated from its attribution. A turn is not simply one restriction or penalty; instead it should be regarded as a first-class entity with attribution.
- **Fast network navigation.** The persisted representation must support fast retrieval of connectivity information for use within network analysis algorithms, and should be structured according to the most common access patterns.

- **Z elevations.** In order to refine network connectivity with planar network datasets (e.g., modeling freeway over and underpasses), logical z elevation values are supplied by commercial data vendors on the ends of each line feature. These elevation values must be respected when establishing network connectivity.
- **Rich attribution of network elements.** To capture real-world constraints, such as one-way travel restrictions, height/weight limits, and time-of-day travel times, we need a rich attribute model that supports multiple attributes on a network element.
- **Uniform attribute access model.** Clients of the model should be insulated from the details of where attribute values originate. For example, the travel time attribute for an edge element may be derived from the properties of the associated street feature, or it may be a real-time value. In each case, client applications should be able to retrieve attribute values without knowledge of the underlying storage.

In addition to these network specific requirements, other standard system requirements such as performance, editability, persistence in a relational database, support for long transactions, and scalability (e.g, a continental dataset of 50+ million edges) also apply.

2.2 Connectivity Model

Connectivity in a network is generally based upon spatial coincidence of the endpoints of line (real-world) features and other point features. This leads to a 1:1 mapping between features participating in a network and the network elements used to represent the network connectivity. This approach works reasonably well for simpler planar network datasets (e.g., TIGER/Line [20], or others commonly available from commercial data vendors such as Tele Atlas or NAVTEQ). However, with non-planar datasets (e.g., long linear features such as highways in transportation networks), it is useful to allow network connectivity partway along a linear feature (we term this *mid-span connectivity*). The familiar one-to-one mapping between linear features and edge elements must be generalized into a one-to-many mapping. Mid-span connectivity is supported in some network models such as the ArcGIS Geometric Network [31]. The example shown in Fig. 1 depicts the one-to-many mapping between line features and edge elements when mid-span connectivity is supported.

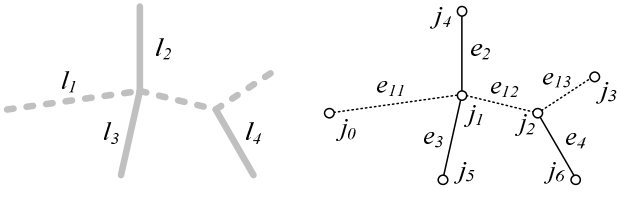


Fig. 1. Mid-span connectivity example; on the left, the long linear feature l_1 (dashed line) will correspond to edge elements e_{11}, e_{12}, and e_{13} if mid-span connectivity is supported

Multimodal Models. As discussed previously, multimodal network models are particularly important in the context of transportation modeling. We employ the concept of connectivity groups within the connectivity model to allow users to group together line classes that should be connected when geometric coincidence is present. A line class may participate in only one group. The number of groups is not constrained. All connectivity is local to a group; line features are not connected to other line features that are found in different connectivity groups. In order to establish connectivity between two groups (e.g., road network in one group, subway network in another group), point feature classes are allowed to participate in one or more groups. Thus, a point feature that is coincident with a road feature in one group and a subway feature in a second group will connect the two groups together in its role as a junction element. Connectivity groups may be employed to model networks containing multiple overlapping subnetworks – e.g., street networks, subway networks, and bus route networks.

An example highlighting connectivity groups is shown in Fig. 2. In this example, the line features participate in two different groups. The first line class contains line l_1 which is depicted by the dashed line. The second line class contains two line features, l_2 and l_3, depicted with solid lines. A point feature class, containing point feature p_1, participates in both connectivity groups. On the right side of Fig. 2, the resulting connectivity is shown. Note that l_1 (edges e_{11} and e_{12}) and l_2 (edges e_{21} and e_{22}) are connected at point p_1 (junction j_3). There is no connectivity between line l_1 and line l_3 as they are in different groups and there is no point feature where they intersect.

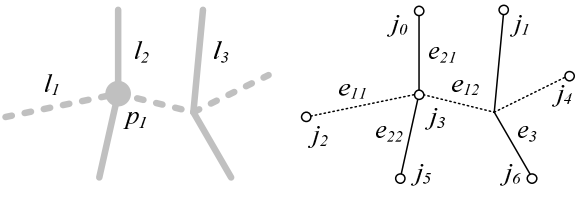


Fig. 2. Example of connectivity groups. Linear feature l_1 (dashed line) is in one group, and features l_2 and l_3 are in a second group. Point feature p_1 is in both groups

Z Elevations. Z elevations (sometimes termed 'z-levs') are a critical component for modeling overpasses and underpasses, tunnels, and highway interchanges with planar datasets (most commercial and governmental datasets are planar). At each endpoint of a line feature, there may be associated z elevation information that is used to refine network connectivity. This elevation information is typically logical – it does not correspond to actual geographical elevations, but rather a logical (ordinal) elevation value. For example, the endpoints of line features representing roads that comprise an underpass may have a z elevation value of 0, while the lines representing the overpass roads may have a value of 1. This logical vertical ordering can extend to support very complex highway interchanges.

Fig. 3 contains an example of four lines meeting at a location that corresponds to an overpass. In the example, lines l_1 and l_3 pass beneath lines l_2 and l_4 (note that all four lines $l_1 - l_4$ share a coincident endpoint; if one did not consider the z elevations when determining network connectivity, all four lines would be connected together). The z elevations are shown (0, 0, 1, and 1 respectively). The resulting connectivity is shown on the right side of the figure. Edges e_1 and e_3 are connected at junction j_{13}; edges e_2 and e_4 are connected at junction j_{24}. Junctions j_{13} and j_{24} appear coincident in the figure.

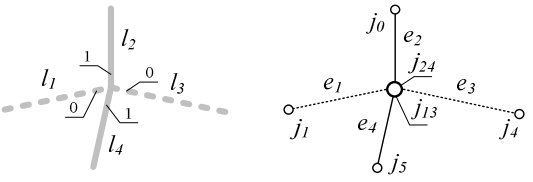


Fig. 3. Example of z elevations and their impact upon connectivity. Z elevations are shown in the left. On the right, the two junctions j_{13} and j_{24} are coincident

The extensions to the simple endpoint connectivity model are reflected in the network build algorithm. During the geometric analysis and connectivity discovery phase of the build process, the connectivity model and the z elevations are used to refine the connectivity between spatially coincident geometries.

2.3 Turns and Maneuvers

Turn restrictions and impedances (or delays) present a problem to most network models. The presence of turns can greatly impact the movement through a network [7], [21]. A common way to model turns within a network is with a turn table [30]. A turn table represents each explicitly specified turn restriction (or penalty) as a row with references to the associated two edges. Turn tables may be augmented with an impedance attribute if the turns may also represent delays or impedances. When traversing the network, the turn table is queried as necessary. An alternative approach is to employ a transition matrix that represents possible transitions at an intersection [10]. The matrix can be encoded into a bitmap for a smaller physical representation.

In order to overcome the performance problems (as perceived by some) of representing turns in an extra table that is disjoint from the network connectivity tables, graph modification techniques have been employed. The goal behind these techniques is to allow the turns to be more directly imbedded within the network connectivity information in order to achieve better performance during network traversals.

Graph Modification – Node Expansion. Node expansion is one technique to imbed turns within a graph by expanding each junction in the graph to a subgraph where permissible turns are explicitly represented as edges [1], [15], [24], [28]. The primary advantage of this approach is that the turns are represented within the connectivity graph of the network (thereby possibly improving traversal performance).

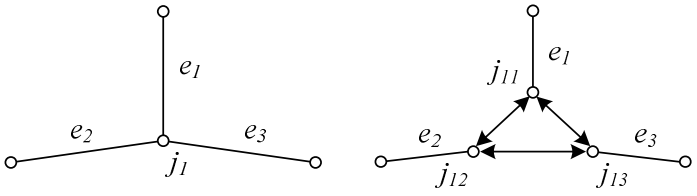


Fig. 4. Example highlighting node expansion where a junction connected to three edges is expanded to a set of three junctions with nine edges representing possible turns (u-turns omitted for clarity)

However, for an intersection of n edges, there are n^2 possible turns (including u-turns). This highlights the fundamental problem with this approach, namely, the significant bloating of the network storage requirements. This adversely impacts both storage costs and traversal performance [21]. In Fig. 4, the intersection junction j_1 is expanded and replaced with three junctions (labeled j_{11} through j_{13}), and edges are used to explicitly indicate permissible turns (the bidirectional edges on the right side of Fig. 4).

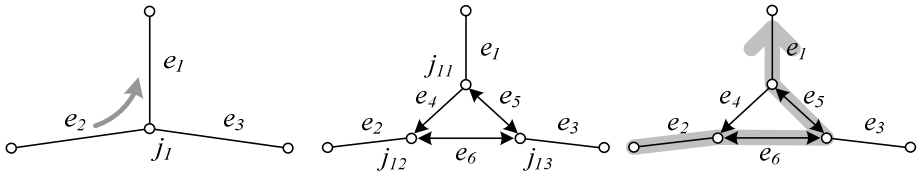


Fig. 5. Example of a turn restriction, the equivalent expanded graph, and an incorrect traversal in the expanded graph

Node expansion also introduces an algorithmic issue caused by traversing the edges in the expanded subgraph in sequence. Such a traversal corresponds to making multiple turns at the same junction in the original graph, and is meaningless. In Fig. 5, the turn from e_2 to e_1 is restricted. The restricted turn is reflected in the expanded graph with the directed edge e_4. However, we can still incorrectly go from e_2 to e_1 via the edges e_6 and e_5. Analysis algorithms that operate on the expanded graph have to avoid such traversals in order to generate correct results.

Graph Modification – Line Graphs. Line graphs (sometimes inappropriately termed dual graphs) are also used to explicitly model turns within a network [2], [30]. Line graphs are a transformation of the original (or primal) graph where edges in the primal are replaced with junctions in the line graph, and edges in the line graph represent turns in the primal. An example is shown in Fig. 6 where a simple (primal) graph consisting of three edges is transformed into the line graph on the right side of the figure. Edge e_1 in the primal is transformed into junction j_{11} in the line graph, edge e_2 into junction j_{12}, and edge e_3 into junction j_{13} respectively. Presuming that all turn movements are allowed, bidirectional edges in the line graph will be created between the three junctions in the line graph (edges e_{12}, e_{13}, and e_{23}).

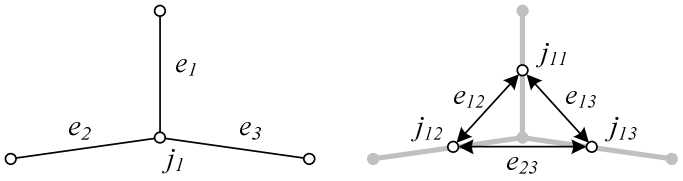


Fig. 6. Example line graph representation. On the right, the primal graph is represented in gray and the line graph is in black. In this example, all turns are possible (u-turns omitted)

Similar to the node expansion technique, the advantage of this approach is that the turns are explicitly represented in the graph. In addition, it results in a smaller graph than with the node expansion technique. However, line graphs require that the primal graph be retained in order to complete certain types of operations such as route drawing [30].

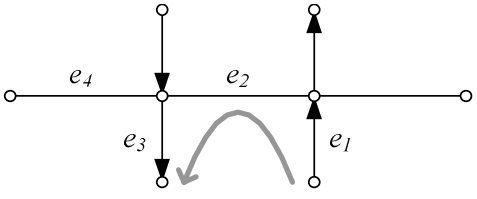


Fig. 7. Example of a three part maneuver e_1-e_2-e_3 at an intersection with a dual carriageway

Maneuvers. A maneuver is a turn that spans three or more edges. Maneuvers are used to model turning movements at complex street intersections within transportation networks. Consider the following intersection formed by a dual carriageway (i.e., a street where each travel direction is represented as a separate line feature) and a two-way street in Fig. 7. To restrict the u-turn from edge e_1 to edge e_3, we need a maneuver composed of the edges e_1, e_2 and e_3 in sequence. The maneuver cannot be synthesized from the two overlapping turns e_1-e_2 and e_2-e_3, since restricting the e_1-e_2 turn also incorrectly restricts the left turn specified by the sequence e_1-e_2-e_4.

Maneuvers can get arbitrarily complicated. We have observed instances of maneuvers with high part counts in transportation networks, such as a nine-part maneuver in the street network for Osaka, Japan. It is awkward to adapt graph modification techniques to model maneuvers.

2.4 Network Attributes

Network attributes are numeric properties of network elements that are used to define the navigation context during an analysis [21]. Examples of common attributes found on network elements include travel time, one-way restrictions, speed along an edge, and hierarchy value. The various types of attributes can be classified as:

- *cost* impedances, which may be apportioned if the line feature is associated with multiple network edges (e.g., travel time),
- *descriptor* a characteristic of the entire element (e.g., speed limits, lane count),
- *restriction* identify which elements cannot be traversed (e.g., one-way), and
- *hierarchy* used in conjunction with hierarchical analysis algorithms (e.g., an order or grade – highways, arteries, and city streets).

Network attributes are usually persisted along with the network elements (e.g., the attribute columns in the standard relational model depicted in Fig. 9). The network attributes often are mapped to attributes found in the associated feature; during the process of building the network and establishing network connectivity, attribute values are read from the features and persisted into the network. The reason for doing so is to minimize the number of tables that must be queried during network analysis, in order to achieve better performance.

However, with very dynamic environments such as are found in location-based service applications, it is sometimes advantageous not to have to persist the value of a network attribute along with the network connectivity. This is particularly the case if the attributes on the feature that are mapped to a network attribute are subject to frequent change. Evaluator components serve to abstract away the underlying storage of the network attributes. Client applications (including the build algorithm and the forward star cursors, as well as analysis algorithms) instead query the evaluators that are associated with attributes and feature classes. The evaluators may return values that are persisted directly in the connectivity network, or they may derive an attribute value on the fly (or even query a web service). In this manner, client applications are presented a uniform view of accessing attribute values.

3 Access Model

There are various approaches to effectively building, maintaining, and navigating the elements contained within a network. Some systems (e.g., [23]) have placed the onus upon the client application for the discovery and maintenance of network elements; client applications are responsible for determining the connectivity and appropriately setting the foreign keys that are used to specify the connectivity in the persisted representation (e.g., setting the from and to junctions on the edge elements). Other previous systems (e.g., [22], [31]) have instead provided mechanisms that perform geometric analysis in order to automatically determine connectivity and persist the information. The choice of when to establish or update the persisted connectivity information is based in part upon the user workflows that the network solution is trying to address.

In addition to building the networks, various approaches have been taken to how the network should be queried. Some systems have relied upon low level querying of the persisted network representation (e.g., which two junctions are connected to the specified edge), while other have provided alternative query mechanisms.

3.1 Workflows

There are two common usage classes among users of network data (from a maintenance standpoint); one class of user purchases their network data (or obtains it from external sources) and infrequently edits or modifies the data. They are instead focused on performing analysis upon the obtained network data. The second class of user is actively engaged in editing and maintaining their data. Most often, the second class of users are large organizations such as the government, utilities, or data providers. We have observed that the first class of user is far more common – most people do not actively edit the features participating in their networks.

For the first class of user that infrequently edits their network data, it is sufficient to support a build process that can complete all geometric and connectivity analysis and network element persistence across the dataset in its entirety. For such users, the network is built immediately following network definition and creation. If the user chooses to edit the features in the network, the entire network will have to be rebuilt in order to guarantee correctness of the connectivity used during network analysis.

With the second class of user that is actively editing the features participating in their network, it may still prove viable to only support a global network build process if the organization can tolerate a build occurring on a periodic basis (e.g., over the weekend; see Section 5 for details concerning building the entire US road network in less than two days). If the organization is editing smaller datasets, the build operation can be staged on a more frequent basis (e.g,, overnight).

There is however a subset of this second class of user (the frequent editors) that needs to have correct network connectivity during the course of editing. For such users, it becomes necessary to support a user-initiated incremental build process where only those portions of the network that correspond to edited features are rebuilt. Techniques may be employed (such as dirty area management with ArcGIS Topology [11]) that will assist in the incremental build of the network.

The need for incremental builds during frequent editing can be obviated if network connectivity is “live”, i.e., network connectivity is automatically re-generated after individual edits to the source data. This alternative approach is used in the ArcGIS Geometric Network [31]; however, it is not as viable here given our rich connectivity model and the complexities introduced by turns and maneuvers.

3.2 Connectivity Queries

For the conventional normalized relational representation, standard SQL queries may be employed. Navigation at this level can prove cumbersome and slow. In some instances, in order to overcome this problem, middleware libraries have been developed [16] that provide analysis functions (e.g., shortest path between two junctions, or the traveling salesman problem [4], [21]). This is useful; however, the navigation is at a high level, precluding clients from developing their own analysis functionality that may require low-level navigation.

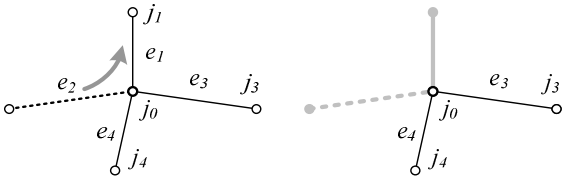


Fig. 8. Example highlighting difference between connectivity and traversability when turn restrictions are present. Connectivity is depicted on the left, with traversability on the right

Alternatively, low-level network navigation may be performed using a forward star adjacency query [6]. A forward star query returns the elements in a network that are immediately reachable from another element. The query is constrained by a set of restrictions (e.g., one-way streets, prohibited turns) that controls which elements are traversable. For example, consider the situation depicted in Fig. 8. In this example, there is a turn restriction at junction j_0 when moving from edge e_2 to edge e_1. This is shown on the left side of the figure. A forward star query at junction j_0 from edge e_2 will result in two edge-junction pairs being returned; namely (e_3, j_3), and (e_4, j_4). The edge-junction pair (e_1, j_1) is not returned as it is not traversable from edge e_2 at junction j_0 because of the turn restriction. From a performance standpoint, forward star queries (and storage representations – see Section 4.2) are the preferred method for querying network connectivity during network analysis operations [26].

4 Physical Model

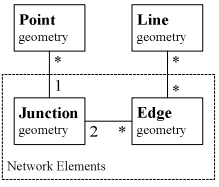
4.1 Standard Physical Implementation

Network topology can be implemented for relational databases in a straightforward fashion as a normalized relational model with explicit representation of network primitives and connectivity using primary and foreign keys (see Fig. 9). This model has been employed in both research and commercial systems [5], [14], [21], [23]. We term this the standard relational network model. A fundamental implementation choice is whether or not the tables representing the network elements (junctions and edges) contain any associated geometry (in Fig. 9, we depict an implementation where geometry is persisted in the network element tables). If geometry is absent from the network tables, they are sometimes referred to as a logical network. If geometry is present, they may be termed spatial networks [16].

The network connectivity is represented by the *from* and *to* junction id foreign keys in the edge table. This representation is definition-based and follows naturally from the mathematical definition of the edges as being a binary relation on the junctions. Attributes may be added to both the junction and edge tables as necessary. It is common to associate impedances or hierarchy values with network elements in this manner.

The normalized relational model is suited to a class of SQL-based connectivity queries. For a given junction (presuming the junction id is known), the connected

edges may be obtained via a selection query of the edge table where either the from or to junction id foreign keys match the specified junction's id value. When traversing a network (e.g, a shortest path computation), each junction that is explored will require a separate SQL query. This can be quite expensive in terms of server loading and suffers from a performance standpoint.



Junction Table

id	geometry	attrs
j_1	point1	…
j_2	point2	…
j_3	point3	…
…	…	…

Edge Table

id	geometry	fr-jn	to-jn	attrs
e_1	line1	j_i	j_k	…
e_2	line2	j_l	j_m	…
e_3	line3	j_n	j_o	…
…	…	…	…	…

Fig. 9. Standard relational network model with geometry represented in the network tables

In order to address this problem, middleware based solutions have been proposed that cache network connectivity information on the client (or application server) and provide access to the information through a conventional API (e.g., Java) on a collection of higher level components [23]. Data management is usually performed via low-level SQL, while navigation and connectivity analysis is via the higher-level API.

A modified adjacency structure is presented in [10] which stores for each edge in the network, a list of possible outgoing edges from its ending junction, taking into account permissible turns between edges. The modified structure does not satisfy our modeling requirements because it only considers turn prohibitions, which are always enforced to constrain the outgoing edges for each incoming edge. In contrast, we regard a turn as a first-class entity with attribution, e.g., one left turn can be used to specify turning restrictions and penalties for different vehicle types using multiple network attributes. Turns do not modify network connectivity, but affect traversability and costs based on the attributes applied during a network analysis. Furthermore, the modified adjacency structure is limited to two-part turns and cannot represent multi-part turns.

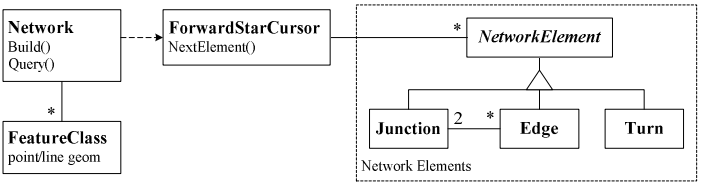


Fig. 10. Basic components in the network engine object model

4.2 Alternative Object Model and Physical Implementation

In order to address some of the problems inherent in the standard network physical implementation and better support the aforementioned requirements (e.g., high performance network analysis functions residing on the client or application server tier)

and workflows, we describe a new object model that is currently hosted within the ArcGIS 9.1 Geodatabase [31]. The basic architecture is a small collection of components that are exposed through industry standard Java, .NET, and COM APIs.

The principal components in the network engine implementation are shown in Fig. 10. The Network is the central component to the system. Chief among its functionality is that which allows the client to build the connectivity of the persisted network representation through geometric analysis of the line and point Features found in the associated FeatureClasses. Each Feature will ultimately correspond to one (or more in the case of line features when the connectivity model is configured to support mid-span connectivity) NetworkElement. The NetworkElement (an abstract class, with three concrete subclasses – Junction, Edge, and Turn) provides an API that allows the direct navigation to the other immediately traversable NetworkElements. It additionally provides a general method for accessing the values of the associated network attributes.

The Network component supports a query model where clients (such as high-performance network analysis algorithms) issue forward star queries [6]. When such a query is issued, a ForwardStarCursor component is returned. This cursor allows the client to index or iterate through the returned traversable NetworkElements (i.e., connected NetworkElements that satisfy the traversability requirements such as respecting turn restrictions, etc.). If necessary, a client can also use lower-level query models supported by the Network component, such as 'give me all NetworkElements that are associated with the specified Feature.'

Storage Representation. The network consists of a collection of tables within a geographical database. The network contains metadata (network definition and connectivity configuration information), junction, edge, and turn elements, the connectivity relationships between them, and the attributes necessary for traversing the network and performing analysis.

Junction Table. The physical storage representation of the network differs, however, from the conventional relational implementation discussed previously (r.e., Fig. 9). While the connectivity information in the conventional implementation is represented as foreign keys within the edge table, we instead represent the connectivity as a set of (edge id, junction id) foreign key tuples that are associated with a junction record in the junction connectivity table. This representation is navigation-based, and is designed to answer the most common adjacency query during network analysis, which is to find the edges and junctions connected to a given junction (r.e., the forward star query in Section 3.2). Each junction record can have four such tuples; if more are needed (e.g., the junction is connected to five or more edges), an overflow table is used.

Junction Table

id	edge1	junc1	...	edge4	junc4
j_1	e_i	j_i	...	e_k	j_k
j_2	e_l	j_l	...	e_m	j_m
j_3	e_n	j_n	...	e_o	j_o
...	...	...	...	...	...

Edge Table

id	from-jn
e_1	j_i
e_2	j_l
e_3	j_n
...	...

Fig. 11. Network storage representation optimized for forward star queries

It is important to note that this storage representation utilizes fixed-length records (r.e., the need for an overflow table). Fixed length records allow us to have direct access into the connectivity data. The number of adjacency tuples in the fixed-length record was determined empirically. For transportation networks, almost all junctions have degree four or less (i.e., the number of edges connected to each junction is typically four or less). This is evident in Table 1, which shows the frequency distribution of junction degrees for a network on Southern California (715,286 junctions and 978,965 edges).

For this network, with four tuples per fixed-length record, the space utilization is 68% with 10 bytes per junction unoccupied or being used for overflow record information. This compares favorably with variable-length records, which would have similar overhead. Only the junctions whose degrees are five or higher (which is less than 0.5% in this network) require two or more records to hold adjacency information; almost all the junctions require only one record for adjacency information. Note that the three-tuple per record representation has a slightly higher space utilization of 76%, but 21% of the junctions would require two or more records.

Table 1. Frequency distribution of junction degrees for a Southern California road network

Degree	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Count	147,86	43,737	375,34	145,40	2,689	234	11	1
Percentage	20.7%	6.1%	52.5%	20.3%	0.4%	0.03	<0.01	<0.01%

Each fixed-length record shown in Fig. 11 is not stored as a row in a relational table; instead, we chose to serialize and compress the rows into larger collections of data (pages) and persist the pages in BLOB tables (an RDBMS column/data type capable of storing binary large objects [13]) within the relational database. The relational database in effect is being used as a paged file system. The network engine components (that reside at either the client or application tiers as described in Section 1 and shown in Fig. 10) provide caching mechanisms and APIs that support both data management and analysis functionality.

Edge Table. The edge table in our storage representation contains the foreign key of the from-junction associated with the edge. If the to-junction is needed, the junction table is queried using the from-junction and edge identifiers. We have observed that finding the from- and to-junctions associated with an edge is actually a fairly uncommon operation during efficient network analysis operations. Thus, we have optimized our storage representation to more effectively support the most common connectivity access pattern - the forward star adjacency query (see Section 3.2). In its simplest form, the forward star adjacency query takes as input a junction element, and returns the set of connected edges and the junctions at the other end of those edges.

Turn Tables. We have chosen not to employ a graph modification technique (e.g., node expansion or line graphs) to represent turning movements as edge elements

within the network. As noted earlier, such techniques are awkward for representing complex turns (maneuvers), and the modified graphs are also difficult to maintain in a dynamic editing environment. Instead, we store turn elements in a turn table, with a representation that is optimized for the most common client access patterns.

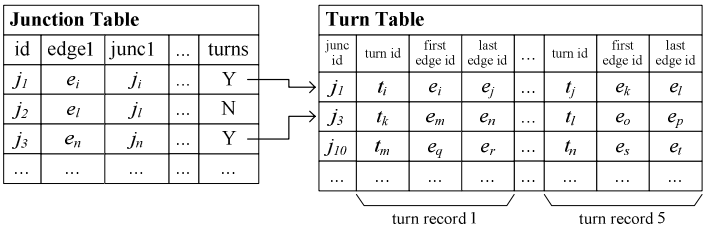


Fig. 12. Turn table representation

The turn table concept that we employ is generalized to effectively support maneuvers as well as the forward star adjacency queries. For each junction in the network, we indicate if there are any associated turns anchored (i.e., the last junction participating in the turn) at the junction. If there are any associated turns anchored at the junction, the turn table contains up to five (turn id, first edge id, last edge id) triplets. An example of a turn table is shown in Fig. 12. If more than five turns are anchored at the junction, an overflow mechanism is supported (similar to that employed with the junction table). As is the case with the junction and edge connectivity tables, we utilize a fixed length record format to facilitate the compression and serialization of the turn table into pages persisted within a BLOB column in the relational database.

During a forward star adjacency query, for a given junction and inbound edge, the turn table may be queried with the specified junction and (first) edge. If an entry matches the (junction, first edge) pair, then the last edge information in the turn entry allows the pairing of the turn with the correct outgoing edge in the forward star result.

4.3 Network Building

As noted earlier, our implementation supports a network building process where the connectivity graph of a network is derived from its source data via geometric analysis. The steps during building are:

1. Extract the geometries of the features in the source data. The extracted coordinates and their feature parentage are stored in a vertex information table.
2. Sort the vertex information table by coordinate values, so that coincident vertices are grouped together.
3. Analyze each group of coincident vertices according to the connectivity model, and generate the appropriate junction elements. During this analysis, vertices that do not connect to other vertices are discarded, while the remaining vertices may be further partitioned into disjoint subsets.

4. Re-sort the vertex information table by vertex, so that vertices from each line feature are re-grouped together.
5. Scan the vertex information table, and generate edge elements connecting adjacent vertices on each line.
6. Analyze turn features and generate associated turn elements.
7. Populate the attribute values of the generated network elements.

Spatial Clustering. When a network analysis algorithm is executing (e.g., a shortest path search between two locations), it typically does not examine the network in a haphazard manner. Instead, there is spatial locality of reference [24]. Areas of the network that are queried next are usually near areas that have already been explored. We exploit this locality by spatially clustering the network elements during the network build process using a space-filling curve (we employed a Peano curve [25]), and persist the network elements in the clustering order within the BLOB pages of the network tables. Other spatial clustering techniques of network elements have been tested and found to be superior to both non-clustered and topologically clustered elements [12].

5 Implementation Experience

This new network model has been implemented and is currently shipping with ESRI's ArcGIS 9.1 product. It addresses each of the requirements enumerated in Section 2.1. It has been used to build very large continent-wide transportation networks, including a dataset derived from the features contained within the entire continental United States (35.9 million line features).

Performance statistics on several different size network build operations are shown in Table 2 (number of linear source features, number of vertices in their geometries, number of network elements created, and the wallclock build time). A reasonable PC (2.4GHz, 2GB RAM) running ArcGIS 9.1 was utilized on the client side and a commercial relational database was employed on the server side. Reported build times include the geometric analysis of the feature geometry in order to establish connectivity, as well as the population of attributes within the persisted network representation (e.g., travel time along an edge). We observed that in the *typical* case, geometry and connectivity analysis consumed 45% of the build process time, while creation of the persistent network elements took 30% of the build time, and population of network attributes the remaining 25% of the processing time.

Table 2. Summary statistics of large networks built

Dataset	*Features*	*Vertices*	*Net Elements*	*Build Time*
U.S. National	35.9 million	128.3 million	65.1 million	43 hours
Northeast U.S.	5.3 million	27.0 million	9.6 million	1.8 hours
Major U.S. Streets	1.8 million	20.4 million	3.1 million	0.5 hours
Paris Metro	0.4 million	0.8 million	0.7 million	< 3 minutes

6 Future Work

There are several areas of ongoing research and development with our network model that will be incorporated into the ArcGIS product following the 9.1 release. These include the direct support for the class of user that is a heavy editor of the features participating in the network (as described in the Section 3.1). In order to support this group, it will be necessary to support the incremental build of the network in the versioned environment. An analogous capability was developed and provided with ArcGIS Topology [11]; this entailed dirty area management, the development of an incremental topology validation process, and incorporation of topology into the ArcGIS Version Management infrastructure. Analogous development tasks will occur with this new network model.

Dirty Areas. A network can have an associated *dirty area* – a dirty area corresponds to the regions within the network extent where features participating in the network have been modified (added, deleted, or updated) but whose connectivity has not been re-established. When the geometry of a feature that participates in a network is modified, the extent of the dirty area is enlarged to encompass the extent of the bounding rectangle of the modified geometry (note that other simplified geometry representations may also be employed - e.g., convex hulls). The dirty area is persisted with the network. In order to ensure that the network is correct, the portion of the network encompassed in the dirty areas will need to be rebuilt.

It is not necessary to build the entire space spanned by the dirty area at one time; instead, a subset of the dirty area can be built. If the dirty area is partially built, the original dirty area will be clipped by the extent of the region that is built.

Allowing users the ability to build a portion of the dirty area is a pragmatic requirement of supporting extremely large seamless network. For example, when a network is first defined, or when the network metadata (e.g., connectivity model, etc.) is modified, the entire extent of the network is dirty. If users were not provided with the capability to build a portion of the dirty area, the user would be required to build the entire network which could prove to be a very lengthy process (e.g., a couple days of processing time for large continent-wide network datasets). As was discussed in [11], the dirty area model effectively supports partial processing in computationally intensive areas of GIS such as topology.

Incremental Build. In order to minimize the amount of processing necessary to maintain a consistent connectivity network, the dirty area mechanism may be exploited in conjunction with an incremental build algorithm. In an incremental build, the connectivity information associated with features in a dirty area is deleted and rebuilt (recreated) in order to achieve a consistent state of the network. The high level algorithm is as follows:

1. Delete the network elements associated with the line features intersecting the dirty area (or portion thereof) being built.
2. Load the geometries of all the line features intersecting the area being built along with the associated network metadata (connectivity model, ternary mapping of evaluators to network attributes and feature classes).

3. Load the geometries of all point features that are connected to the line features intersecting the area being built (these point features may lie outside the area being built if the line feature extends outside the area).
4. Sort the vertices associated with the features, retaining the parentage information.
5. Discard all isolated line vertices (i.e., the vertices on the line features that are not coincident with other vertices from differing line or point features).
6. Discard interior line vertices if the connectivity model does not support mid-span connectivity on the associated line feature class.
7. Create network junctions as necessary for the remaining line vertices and other isolated vertices associated with point features.
8. Create network edges between the junctions as appropriate based upon the connectivity model.
9. Update the dirty areas associated with the network.

It is important to note that the network build process does not need to span all features within the network. A build can be performed on a subset of the space spanned by the dataset. This is a complex task since the re-built portion of the network has to be properly stitched together with the rest of the network.

7 Conclusion

In this paper we described the logical model of GIS network topology and several extensions to the standard network model that directly facilitate the modeling of multimodal systems, supporting mid-span connectivity on line features, as well as supporting endpoint elevation data that often accompanies large planar datasets from commercial data vendors. We reviewed a common physical database implementation that uses the conventional notions for mapping entities and relationships to tables and the standard primary key / foreign key referential integrity model. Problems with this approach were discussed. We then presented an alternative implementation of the network model which used a different physical approach to persisting network connectivity. This new model additionally supports turn restrictions and impedances, both two-part turns as well as multi-part turns (maneuvers). Efficient mechanisms for navigating the network connectivity were discussed (the forward star adjacency query), as well as a more flexible mechanism (network evaluators) for maintaining and querying attributes on the network elements. This design serves as the basis for our implementation of transportation networks in the ArcGIS geographic information system; this new model has been implemented and is currently shipping with the ArcGIS 9.1 product.

Our future work will focus on extending the network with support for dirty area management policies and an incremental build algorithm that is more useful for organizations that frequently edit their network data (e.g., governmental organizations and commercial data providers). In addition, we will be supporting this network model in the distributed database environment, incorporating aspatial features (features without geometry) into the network model, as well as other performance enhancements.

Acknowledgements

Numerous other individuals within ESRI Development were involved in the design and implementation of this network model in the ArcGIS 9.1 product. Key people included Frederic Albert, Gillian Allen, Hal Bowman, Jai Chakrapani, Matt Crowder, Craig Gillgrass, Alan Hatakeyama, Rich Krabill, Jim McKinney, Sudhakar Menon, Scott Morehouse, Jay Sandhu, Frederic Schettini, Doug Sterling, and Jeff Wickstrom.

References

1. J. Añez, T. de la Barra, and B. Pérez. Dual Graph Representation of Transport Networks. *Transportation Research* 30(3), June 1996
2. T. Caldwell. On Finding Minimum Routes in a Network with Turn Penalties. *Communications of the ACM* 4(2), February 1961
3. Caliper. *TransCAD: Transportation GIS Software Ref. Man.*, Newton, MA, 1996
4. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms, 2nd Edition. MIT Press, Cambridge, Massachusetts, 2001
5. K. Dueker and J. Butler. *GIS-T Enterprise Data Model with Suggested Implementation Choices.* PR101, Center for Urban Studies, Portland State, 1997
6. J. Evans and E. Minieka. *Optimization Algorithms for Networks and Graphs.* M. Dekker (editor), New York, 1992
7. ESRI. *Network Analysis; Workspation ARC/INFO version 8.1*. Prepared by Environmental Systems Research Institute, Redlands, California, 2001
8. T. Foresman (editor). *The History of Geographical Information Systems*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998
9. M. Goodchild. Geographic Information Systems and Disaggregate Transportation Planning. *Geographical Systems* 5, 1998
10. R. Güting, V. de Almeida, and Z. Ding. *Modeling and Querying Moving Objects in Networks*. FernUniversität in Hagen, Informatik-Report 308, 2004
11. E. Hoel, S. Menon, and S. Morehouse. Building a Robust Relational Implementation of Topology. In *Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases (SSTD 2003).* Santorini Island, Greece, July 2003
12. Y.-W. Huang, N. Jing, and E. Rundensteiner. Optimizing Path Query Performance: Graph Clustering Strategies. *Transportation Research Part C 8*, 2000
13. International Organization for Standardization (ISO). *ISO International Standard: Database Language SQL – Part 2: Foundation (SQL/Foundation)*, ANSI/ISO.IEC 9075-2:99, September 1999
14. C. Jensen, T. Pedersen, L. Speičys, and I. Timko. Data Modeling for Mobile Services in the Real World. In *Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases (SSTD 2003)*. Santorini Island, Greece, July 2003
15. R. Kirby and R. Potts. The Minimum Route Problem for Networks with Turn Penalties and Prohibitions. *Transportation Research* 3, 1969
16. R. Kothuri, A. Godfrind, E. Beinat. *Pro Oracle Spatial*. Apress, Berkeley, 2004
17. L. Lang. *Transportation GIS*. ESRI Press, Redlands, California, 1999
18. P. Longley, M. Goodchild, D. Maguire, and D. Rhind (editors). *Geographical Information Systems, Volume 2*. John Wiley & Sons, New York, 1999

19. M. Mainguenaud. Modeling the Network Component of Geographical Information Systems. *International Journal of Geographic Information Systems* 9(6), 1995
20. R. Marx. The TIGER System: Automating the Geographic Structure of the United States Census. *Government Publications Review* 13, 1986
21. H. Miller and S.-L. Shaw. *Geographic Information Systems for Transportation.* Oxford University Press, Oxford, England, 2001
22. S. Morehouse. ARC/INFO: A Geo-Relational Model for Spatial Information. In *Proc. of the 7th Intl. Symp. on Computer Assisted Cartography (Auto-Carto 7)*, Washington, DC, March 1985
23. Oracle. *Oracle Database 10g Network Data Model*. Prepared by Oracle Corporation, Redwood Shores, California, 2003
24. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *Proc. of the 29th VLDB Conf. (VLDB 2003)*. Berlin, 2003
25. G. Peano. Sur une Courbe Qui Remplit Toute une Aire Plaine. *Mathematische Annalen*, 36: 157-160, 1890
26. B. Ralston. GIS and ITS Traffic Assignment: Issues in Dynamic User-Optimal Assignments. *GeoInformatica* 4(2), June 2000
27. P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann, San Francisco, 2002
28. L. Speičys, C. Jensen, and A. Kligys. Computational Data Modeling for Network-Constrained Moving Objects. In *Proc. of the 11th ACM Intl. Workshop on Advances in Geographic Information Systems (ACM-GIS'03)*, New Orleans, Nov. 2003
29. F. Southworth and B. Peterson. Intermodal and International Freight Network Modeling. *Transportation Research Part C 8*, 2000
30. S. Winter. Modeling Costs of Turns in Route Planning. *GeoInformatica* 6(4), 2002
31. M. Zeiler. *Modeling Our World: The ESRI Guide to Geodatabase Design*. ESRI Press, Redlands, California, 1999