Geoinformatica (2008) 12:219–253
DOI 10.1007/s10707-007-0032-1

Enabling Location-based Services—Multi-Graph Representation of Transportation Networks

Laurynas Speičys · Christian S. Jensen

Received: 2 December 2006 / Revised: 18 May 2007 /
Accepted: 29 May 2007 / Published online: 17 August 2007

Abstract Advances in wireless communications, positioning technologies, and consumer electronics combine to enable a range of applications that use a mobile user's geo-spatial location to deliver on-line, location-enhanced services, often referred to as location-based services. This paper assumes that the service users are constrained to a transportation network, and it delves into the modeling of such networks, points of interest, and the service users with the objective of supporting location-based services. In particular, the paper presents a framework that encompasses two interrelated models—a two-dimensional, spatial representation and a multi-graph presentation. The former, high-fidelity model may be used for the positioning of content and users in the infrastructure (e.g., using map matching). The latter type of model is recognized as an ideal basis for a variety of query processing tasks, e.g., route and distance computations. Together, the two models capture central aspects of the problem domain needed in order to support the different types of queries that underlie location-based services. Notably, the framework is capable of capturing roads with lanes, lane shift and u-turn regulations, and turn restrictions. As part of the framework, the paper constructively demonstrates how it is possible map instances of the semantically rich two-dimensional model to instances of the graph model that preserve the topology of the two-dimensional model instances. In doing so, the paper demonstrates how a wealth of previously proposed query processing techniques based on graphs are applicable even in the context of complex transportation networks. The paper also presents means of compacting graphs while preserving aspects of the graphs that are important for the intended applications.

L. Speičys (✉) · C. S. Jensen
Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7E, DK-9220, Aalborg, Denmark
e-mail: laurynas@cs.aau.dk

C. S. Jensen
e-mail: csj@cs.aau.dk

Keywords location-based services · multi-graph model · transportation network model · spatial network model · lane-based model

1 Introduction

Continued improvements in the affordability and functionality, and thus popularity, of Internet-worked mobile devices such as mobile phones, personal digital assistants, and navigation systems enable a range of new personal information services, many of which will exploit information about the user's geo-location for providing the desired functionality. Example services include applications that allow mobile phone users to locate friends or family, businesses, or landmarks. Services may also deliver maps, directions, or traffic reports.

One may distinguish among three location-based service scenarios depending on the assumed degree of freedom with which the service users can move. The first occurs when the users, e.g., sailors at sea, are considered capable of unconstrained movement in the relevant physical space. The second occurs when user movement is partially constrained due to obstacles. For example, the movement of a cross-country jogger is constrained by a fence. The third scenario occurs when user movement is restricted to a network-type structure, the prototypical example being users in cars that move in a road network.

This paper concerns the third scenario, which is applicable to many kinds of mobile services, e.g., those that supply moving users with information about relevant, stationary objects. The paper addresses the modeling of transportation networks at the level of lanes, and it also considers the modeling of points of interest and service users.

The contributions are three-fold. As the first contribution, the paper presents a framework that consists of a two-dimensional, geographical model and a derived graph model of road networks. While the modeling of real-world phenomena such as transportation networks is an open-ended process where additional aspects can always be captured, care has been taken to capture aspects that relate to the topology of transportation networks. Thus, the framework captures aspects such as the following.

- bi-directional as well as one-directional roads
- lanes and the associated lane-change regulations
- u-turn regulations along roads
- turn regulations, such as no left turn and no u-turn, at intersections
- travel distances and travel times, e.g., as caused by road conditions

This type of detailed lane-level modeling of transportation networks is becoming increasingly relevant because advances in positioning technologies are slated to enable the positioning of vehicles within lanes. For example, infrastructures that rely on in-road and in-vehicle sensors for accurate positioning at lane resolution are being conceived in the telematics community.

Positioning at lane resolution will increase the quality of existing services, e.g., navigation services, but will also enable entirely new services, e.g., collision warning and assisted driving services.

In addition, the framework captures points of interest in relation to the transportation network models. Points of interest may have multiple accessibility points at lane resolution. The framework also captures movement-capable objects, termed service users.

The second contribution is to leverage the large body of existing query processing techniques based on graphs. The paper demonstrates that, and how, this body of work is relevant for location-based services that rely on accurate modeling of complex transportation networks.

Specifically, a central component of the paper's framework is a mapping of instances of the semantically rich two-dimensional model to instances of the graph model. This mapping demonstrates how it is possible to represent complex transportation network by semantically simple graphs. The graph model and mapping are built to accurately capture the topologies of transportation networks while also serving as a foundation for query processing. To enable distance computations, the graph model includes edge weights; to capture roads with multiple lanes, the model is a multi-graph; and to capture the ability to make u-turns in-between intersections (in-between graph vertices) and the movement from one lane to another, so-called co-edge and change-edge relations are also included.

Third, the paper presents techniques aimed at compacting graph model instances without changing their semantics as seen by their intended applications. Different intended uses of a graph generally pose different requirements to the fidelity of the graph. For example, shortest path search does not rely on the accurate representation of multiple lanes between pairs of vertices if these edges have equal weights; so with shortest path search being the objective, multiple edges between the same pair of vertices can thus be reduced to one edge in such cases. The paper takes as its outset several common uses of graphs in location-based services and then presents transformations that compact graphs while preserving the detail relevant for these uses.

It is expected that most of the query processing underlying location-based services will occur in the graph model and that the two-dimensional model will mainly serve as a basis for the positioning of the points of interest and moving service users. Therefore the paper's focus is on the graph model and the mapping to this model. Other two-dimensional models than the one utilized in this paper may well be equally suitable.

2 Related work

Within computer science, past research has covered the efficient support for a variety of types of queries, including different types of range queries, nearest neighbor queries, and reverse nearest neighbor queries (e.g., [3], [18], [28], [31], [33]). However, this line of research generally makes simple assumptions about the problem setting. Perhaps most prominently, much work assumes that data and mobile objects are points embedded in, typically, two-dimensional Euclidean space (e.g., [11], [16]). The resulting notion of proximity renders the techniques inadequate for many location-based services. This paper's proposal enables the use of the notion of road distance, which is often the relevant notion of distance. Some recent proposals for query processing techniques do assume road networks; we discuss these shortly.

Other scientific work has considered problems where the infrastructure is central. The classical approach is to use either non-directed or directed graph for the modeling of road network. Both are compact mathematical representations that capture some essential properties of roads. An exemplary problem addressed in this setting is that of finding a shortest path between locations [4], [7]. In more recent work, graphs have been used for the processing of nearest neighbor queries for moving objects [26].

More recent approaches use graphs with different spatial embeddings, sometimes called *spatial network models*, for the modeling of transportation networks.

In one line of work [19], Voronoi diagrams are built over spatial network with the purpose of efficient query processing. In this research, a spatial network is a weighted graph consisting of vertices and weighted edges. Vertices, which have coordinates in the two-dimensional plane, are created for connection points where roads meet and for points of interest. Network Voronoi diagrams are then created based on the vertices that correspond to data points, so that each such vertex has an associated Voronoi region in two-dimensional space. The proposal uses R-tree indexing of these regions.

Another line of research [1], uses graphs in conjunction with variations of Dijkstra's algorithm. Here, a spatial network is a weighted, directed graph. In addition, each edge has an associated polyline that captures the location in two-dimensional space of the road modeled by the edge. In this line of research, R-trees are used for the indexing of the edge polylines. Data points are located on the edges—the position of a data point is given by an edge and a distance from the start of that edge.

In a different line of research on query processing in spatial networks [25], a spatial network is a weighted, directed graph where additional spatial components are associated with the edges; specifically, each vertex, which models a road intersection, is given a point position in two-dimensional space, and the weight of an edge, which models a road segment, captures the length of (or travel time associated with) the segment.

In yet other work, spatial networks capture an embedding into Euclidean space and a conventional graph in a unified fashion. These models go further than the ones just covered by also considering the modeling of turn restrictions at intersections and the modeling of both one-way and bi-directional roads [15], [20], [23].

We note that none of these approaches capture a road infrastructure at the level of detail of lanes, where issues such as the abilities to change lane and make u-turns must be captured.

Next, in contrast to the research community's contributions that aim to enable the efficient processing of advanced and specialized spatial queries, solutions offered by industry are generic and support the processing of basic spatial queries [22]—these contributions use geographically embedded geometries as their primary data structure and do not capture graphs.

In the domain of Geographic Information Systems for Transportation [14], focus has been on the development of database models for road-related data [8]. These typically use some form of linear referencing (e.g., [21]) for the capture of data object locations within road networks. The main objective has been to provide foundations for integrating different road-related content relevant to administrative tasks, not location-based services for mobile users.

Navigable data models have been created to support a multitude of tasks including vehicle navigation. We are aware of only a few works that have taken it as a requirement to devise models that capture road network details such as lanes and the connectivities among lanes. Planar [13] and non-planar [10] representations have been suggested, where the latter significantly improve on data maintenance. The non-planar model captures the geo-location and topology of lanes. The topological information consist of lane connectivities, of turn restrictions along lanes and at points, and of impedance points. This information can then be used for constructing a graph on which a search along the lanes can be performed. This paper extends this line of work by constructively demonstrating how instances of such models can be mapped to instances of an appropriate graph model. It also maps points of interest and moving services users to these instances.

It should also be noted that modern "multi-purpose" Geographic Information Systems [32], transportation-oriented systems [17], and an increasing number of systems that offer route planing are related to this paper's contributions. Such systems rely on models of road networks for their functioning. However, these models are generally proprietary and not available to the research community, or they appear limited in their fidelity (e.g., they ignore lanes) and their intended uses, in comparison to our proposal. This paper provides an expressive computational data model for location-based services that is open to the research community, so that further research may be built upon this model.

The paper is outlined as follows. Section 3 further elaborates on the addressed problem. The two following sections present the two-dimensional and the graph models of road networks and of static and moving data objects. This is followed by a formal description of the transformation of the two-dimensional representation to the graph representation in Section 6. Transformations that enable making graphs more compact are presented in Section 7. Finally, Section 8 concludes and offers directions for future research.

3 Application scenario

Our fundamental objective is to support mobile services that exploit location information from the service users to deliver the desired functionality.

We assume that the movements of the service users are restricted to a road network; and we assume the presence of two kinds of objects: moving and static. The moving objects, which we term service users, are capable of continuous movement. The static objects, all of which may be reached via the road network, are the objects of interest to the service users, e.g., hospitals, gas stations, and exhibitions. We also assume that the moving objects are on-line and may communicate their location to a central server.

We anticipate that lane-level positioning will become possible and thus aim to support this level of accuracy in the framework. This positioning may be accomplished via the Global Positioning System or the emerging Galileo positioning system, via the wireless service infrastructure, via a combination of these, or via other means such as in-road sensors. In particular, current radio positioning technologies are capable of positioning accuracies of approximately 1 m. These technologies are close to being sufficient for lane-level positioning on highways.

According to the U.S. Department of Transportation, the overwhelming majority of highways in the U.S. have lanes that are approximately 3.7 m (12 ft) wide [29]. A receiver that is located at the center of such a lane and that is capable of providing positioning with an accuracy of 1 m is very likely to allow correct lane identification. This renders several, already existing technologies [2], [30] capable of providing sufficiently accurate positioning for lane-based navigation in highway conditions. The positioning in conditions different from highways—on roads with narrower lanes, roads in urban areas, etc.—may be more challenging. However, other, emerging positioning technologies [9], [12], [24] strive to provide positioning accuracies that are similar or greater to the one described even under these more challenging conditions.

Next, radio positioning enhanced with information from various in-road sensors hold the potential for enabling positioning at lane-level accuracy for lanes of any width.

Finally we note that in practice, the accuracy of positioning depends also on the map matching algorithms that map positions emitted by a positioning device to a position in the road network.

The assumed setting allows the moving objects to use services that involve various location-related queries about the objects in the road network, e.g., range queries, k nearest neighbor queries, and reverse nearest neighbor queries, in addition to active, continuous, and ranked (ordered) versions of these. For example, a user may issue a query such as this: "display an up-to-date list of the three nearest, open pharmacies within 6 km."

To enable the processing of queries, appropriate representations of road networks and moving and static objects are needed. As the moving objects are restricted to the road network, the distances between pairs of objects have to be expressed in terms of the road network—Euclidean distances do not suffice.

When designing a data model for representing this data, two properties attract special attention, namely the efficiency with which queries may be processed against the representation and the fidelity with which the mini-world may be captured using the representation. Often, one must be traded for the other. A simple, low-fidelity representation may be most amenable to efficient query processing. When increasing the fidelity, more detail is captured, which leads to a more complex and voluminous model that is less efficient for querying. To obtain both expressiveness and efficient support for queries, we use two complementary and interrelated models, a two dimensional and a graph model.

As an illustration, consider Fig. 1, which depicts a sample road network that embodies a few of the aspects that the two representation must be capable of capturing. This network illustrates some of the configurations of lanes possible in real road networks. In particular, the network includes: bi-directional roads with several lanes in each direction, the disappearance and an appearance of a lane due to local traffic access to a highway, and the abrupt disappearance of lanes, i.e., a bi-directional road turning into a single-directional road without reaching an intersection and a road splitting into two.

The figure also illustrates different lane change restrictions. In the residential areas to the right, there are no restrictions on lane changes and u-turns. However, u-turns are not allowed on the bridge that crosses the highway (top right), and on the highway, lane changes are prohibited from the access lane for local traffic.

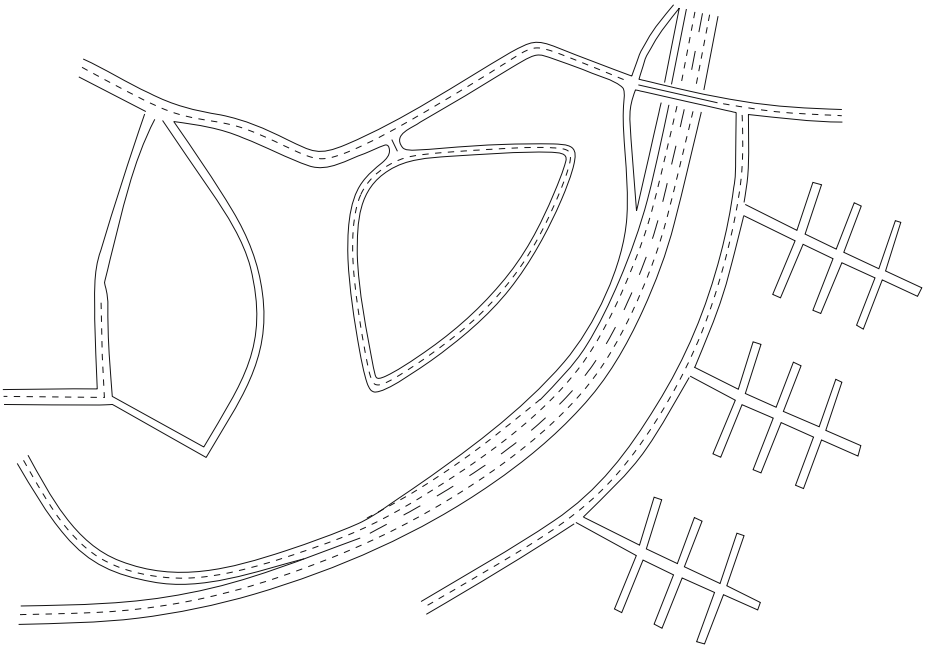


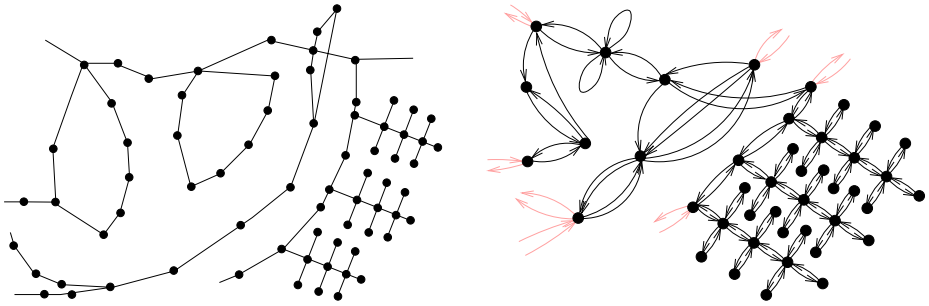
Fig. 1 Example road network

Figure 2a and b illustrate the two-dimensional and graph representations of this network.

The first representation is two-dimensional in that it captures the approximate geographic locations of the road network elements. A third dimension capturing height can be easily added. However, as this adds little in the way of inherently new challenges within this paper's topic, we consider only two dimensions. This representation, which is capable of capturing extensive detail, consists of line segments that represent (parts of) roads. This type of representation is required for our scenario. Because the locations of mobile objects may be provided in (an equivalent of) Euclidean coordinates, a correspondence between the roads and their locations in Euclidean space is necessary in order for us to be able to place the moving objects in the road network.

The bulk of the query processing occurs within the second representation, which is a directed, weighted multi-graph. The graph is a multi-graph because it may contain "duplicate" edges. In comparison to the two-dimensional representation, this graph representation of a road network is completely separate from the geographic space into which the road network is embedded.

It captures the topology of the road network. Put differently, it captures the connectivity offered by the road network, i.e., the movement possible within the network. Specifically, edges represent lanes in-between intersections along with their movement directions, and they also capture the allowed movements at intersections. Edge weights capture properties that influence movement. The graph representation



a Two-Dimensional Representation **b** Graph Representation

Fig. 2 Two-dimensional and graph representations of road networks

also includes relations that capture the possibilities of u-turns and lane changes outside of intersections.

In keeping with its objective of being a computationally efficient basis for query processing, the graph representation is a more abstract, structurally simple, and compact structure than the two-dimensional one. The structural simplicity is due to its use of only a few modeling constructs, each of which is very simple.

To summarize, the paper proposes to use two complementary representations for supporting query processing in relation to location-based services. A two-dimensional representation enables us to position geo-referenced objects in the road infrastructure, and a graph representation is used for most of the actual query processing. The actual uses of these representations in query processing is beyond the paper's scope.

Precise descriptions of the two representations and of a procedure for building a graph representation that corresponds to a two-dimensional representation are given in Sections 4, 5, and 6.

4 Two-dimensional representation

We begin the description of the framework by defining the two-dimensional (2D) representation. In particular, we show how traffic regulations are captured.

The 2D representation captures the (approximate) geography of a road network, of static and moving objects, as well as all other relevant data. It serves as the basis for forming the subsequent graph representation, which is built as a transformation of the 2D representation. The 2D representation and the transformation are utilized when mapping data into the graph representation, since majority of locations are expected to be given to us as geographical (Euclidean) locations.

4.1 Road networks

At the most abstract level, the 2D representation of a road network is given by a two-tuple $RN^{2D} = (S, C)$, where S is a set of segments and C is a set of connections; we consider these two elements next.

4.1.1 Segments

Segments capture the approximate geography of roads. We capture a road by a zero-width curve that represents the centerline of the road. This curve is represented as a sequence of connected line segments where each line segment is given by a pair of delimiting points. Such a sequence, which is called a polyline, can approximate arbitrarily complex shapes of roads through adjusted lengths of their segments. The combination of the delimiting points and the positions along the polyline segments capture the embedding into geographical space of a road.

Segments also capture the traffic regulations that are associated with lanes. We capture the division of roads into lanes and the associated horizontal demarcation indicating the allowed movement from one lane to a neighboring lane. Examples of these are markings in-between two lanes that prohibit movement between the two lanes; that prohibit movement from one lane to the other, but not vice-versa; and that do not restrict the movement (this can also refer to the absence of an explicit demarcation, while the configuration of lanes is implied by other means, e.g., by traffic signs).

We also capture traffic regulations that affect the movement of objects while they stay within a lane. These may include speed limits, zones of increased danger, e.g., a zone with ongoing pavement repairs. These regulations are captured as all other movement-affecting properties. We capture properties that affect movement if this effect can be quantified. For example, the properties may be: the average traffic congestion, the pavement wear-and-tear state, and a personal preference. These, and a whole spectrum of other properties, can be quantified as conditions that hinder or facilitate movement with respect to some nominal movement condition. The effects of such properties are normalized over the network.

We proceed with a more formal definition of segments. A *road segment* s, or simply a *segment*, is a four-tuple $(p_s, p_e, l, prop)$. The first two elements belong to $\mathbb{R}^2$ and are the start and end points of the segment. Element $l = ((l^-, l^+), v)$ captures the number of lanes in each movement direction and the connectivity between neighboring lanes, i.e., whether it is allowed to change from one lane to another. The last element is a set of properties that affect the movement along the lanes.

More specifically, $p_s = (x_s, y_s)$ and $p_e = (x_e, y_e)$, where $p_s \neq p_e$, jointly denote the delimiting points of s, and they specify location of all points p of the segment. Thus, $p \in s$ iff $p \in \{a(x_e - x_s, y_e - y_s) + (x_s, y_s) \mid a \in [0, 1]\}$.

Next, we capture the lanes of the segment by the element $l = ((l^-, l^+), v)$, where the absolute value of $l^- \in \mathbb{Z}_0^-$ denote the number of lanes in the *es* direction and $l^+ \in \mathbb{Z}_0^+$ denotes the number of lanes in the *se* direction. We identify the lanes of a segment by numbers, as follows. Lanes that allow movement from p_s to p_e, defined as the *se direction*, are numbered $1, 2, 3, \ldots, l^+$, with the numbering starting at the leftmost lane (the lane closest to the opposite movement direction). Similarly, the lanes that allow movement from p_e to p_s, defined as the *es direction*, are numbered $-1, -2, -3, \ldots, l^-$. The value 0 is used for l^+ and l^- to capture the absence of a traffic direction. For later use, we define the domain of all lane numbers for segment s as $\mathbb{Z}^s = \{i \in \mathbb{Z} \mid l^- \leq i \leq l^+ \wedge\ i \neq 0\}$.

Next, the set of pairs of lanes v captures the lane-change restrictions on a segment, by capturing the changes between neighboring, or adjacent, lanes that are possible. The change from the leftmost lane in one direction to the leftmost lane in the other

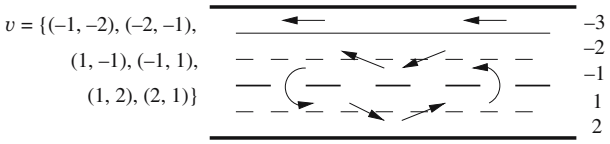


Fig. 3 Road segment with lanes and lane-change restrictions

direction, represents a u-turn. Set v thus contains the tuple (i, j) if a change from lane i to lane j is allowed. Note that the changes between lanes can be asymmetric. Lanes i and j are adjacent if $j = i \pm 1$ or $i, j \in \{-1, +1\} \wedge i \neq j$. Lane changes between non-adjacent lanes are inferred from v by transitivity.

Example 1 Consider the example road segment in Fig. 3. The segment has two lanes in the *se* direction and three lanes in the *es* direction. The lane numbering is indicated along with example lane-change restrictions. Specifically, u-turns are allowed from both travel directions, as are lane changes between the two first lanes (i.e., numbers closest to 0) in each direction. We may imagine that the rightmost lane in the *es* direction is for local traffic. In the figure, we thus have $l^+ = 2$, $l^- = 3$, and $v = \{(1, 2), (2, 1), (1, -1), (-1, 1), (-1, -2), (-2, -1)\}$.

Finally, *prop* is a set of (*property*, *value*, *lanes*) tuples that capture properties that affect the movement along a segment. The first element, *property* identifies a property. The second element *value* belongs to $\mathbb{R}^+$ and quantifies the effect on the movement along the segment that the property has. Example of such properties include an average traffic load and a driver's personal preference. The intuition is that the length of a segment is multiplied by the property values that apply to the segment. Values between 1 and ∞ then slow down movement (they make the segment "longer"), and values between 0 and 1 speed up movement (they make the segment "shorter"). The third element specifies the set of lanes on the segment for which the pair of a property and a value hold.

Example 2 Consider the 2D representation in Fig. 4. Table 1 provides information about four segments from the figure, each of which has one lane in each movement

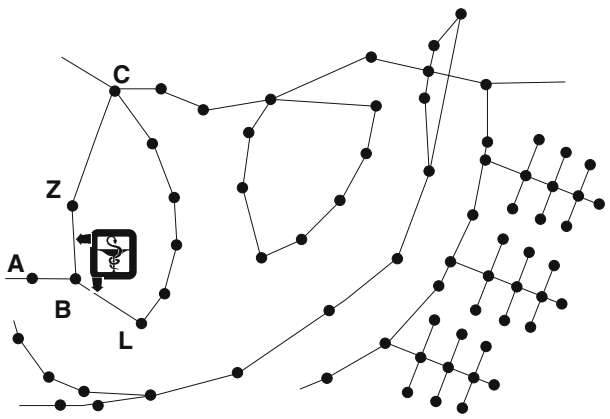


Fig. 4 2D representation of a road network

Table 1 Example segments

s	p_s	p_e	l	$prop$
AB	(4639,2481)	(4714,1925)	$((-1,1),\{(-1,1),(1,-1)\})$	$\{(pr_{sp}, 0.67, \{-1,1\})\}$
BZ	(4714,1925)	(4717,1599)	$((-1,1),\{(-1,1),(1,-1)\})$	$\{(pr_{sp}, 1.5, \{-1,1\}),$ $(pr_{bmp}, 1.1, \{1\})\}$
ZC	(4717,1599)	(4906,1086)	$((-1,0),\emptyset)$	$\{(pr_{sp}, 1, \{-1\})\}$
BL	(4714,1925)	(5128,1822)	$((0,1),\emptyset)$	$\{(pr_{sp}, 1.2, \{-1\}),$ $(pr_{dmg}, 1.1, \{1\})\}$

direction. The property pr_{sp} captures the speed limits of the segments (in km/h). The effect of the speed limit is normalized with respect to a nominal value of 60. Thus, the speed limit is 90 on segment AB, 40 on BZ, 60 on ZC, and 50 on BL. Additionally, the table indicates that segment BZ has several bumps in the *se* direction and that BL has damaged pavement. These are represented by properties pr_{bmp} and pr_{dmg}, respectively.

4.1.2 Connections

Connections are structures that connect segments. Each connection has a geographical position, captures which segments it connects, and records also the traffic regulations that are imposed between lanes of the segments it connects.

The geographical location allows us to identify location of the physical entity, e.g., an intersection, corresponding to a connection. In addition, the location may be helpful in identifying the connected segments, by using the proximity of their delimiting points to the location.

The set of connected segments captures the topology of the network. For example, the set can indicate that two roads are connected although the delimiting points of their corresponding segments do not coincide, or, conversely, it can indicate that certain roads are *not* connected although the delimiting points of their corresponding segments do coincide, e.g., on roads with lanes where one is above the other.

The traffic regulations imposed at a connection indicate the allowed movement to/from a lane from/to other lanes on the connection.

In more precise terms, a connection c is a four-tuple (p, S^c, mx, id). The first element p belongs to $\mathbb{R}^2$ and denotes the geographical point location in two-dimension space of the connection. A connection is typically located near or exactly on the ending point(s) of one or more segments. As an example, consider Fig. 4, where black dots represent connections. Some connections are labeled by capital letters A, B, C, Z, and L.

The second element of a connection captures the set of segments that meet at the connection. The segments that meet at connection c are referred to as the *connection segments* of c and comprise the set S^c. For simplicity, we require that all connection segments of a connection be delimited by the point location of that connection. This requirement can be substituted with a requirement for a certain proximity to the point location of c, e.g., it may be required that delimiting points of connection segments are within a certain radius of a connection at which they meet. In the figure, the connection with $p = \text{B}$ has {AB, BL, BZ} as its second element.

The third element mx of a connection is a *connection matrix*. It specifies for every pair of a lane i of a segment $s \in S^c$ and a lane j of a segment $s' \in S^c$ whether it is allowed to move from lane i to lane j Note that s and s' need not be different segments. The matrix is given as a set of tuples, where a tuple indicates that it is allowed to move over the connection from the lane in the first element of the tuple to the lane in the second element, i.e., $mx = \{(s, i), (s', j) \in (S^c \times \mathbb{Z}^s) \times (S^c \times \mathbb{Z}^{s'}) \mid$ it is allowed to move from lane i of segment s to lane j of segment $s'\}$.

Connection matrices enable the capture of the specific traffic regulations that apply at a connection. For example, it can capture prohibited/enforced turns at intersections, allow u-turn at a separating line, indicate allowed transitions between lanes at intersection areas where a road is widening or narrowing, or capture the transitioning of lanes at places of splits and merges of roads.

Finally, the fourth element id is a unique value that is included to allow us to reference connections.

Example 3 Consider the connection labeled B in Fig. 4. A connection matrix for the connection is $\{((\text{AB}, 1), (\text{BZ}, 1)), ((\text{AB}, 1), (\text{BL}, 1)), ((\text{BZ}, -1), (\text{AB}, -1))\}$. It indicates that it is allowed to move from the *se* lane of AB to the *se* lanes of BZ and BL. It is also allowed to move from the *es* lane of BZ to the *es* lane of AB. However, a left turn from BZ to BL is not allowed, i.e., $((\text{BZ}, -1), (\text{BL}, 1)) \notin mx$.

The regulations at a connection can have many configurations. However, every configuration must be consistent: (1) for each lane that allows traffic movement into a connection, there must be at least one lane that accepts the incoming traffic, and (inversely) (2) for each lane that allows traffic movement out of the connection, there must be at least one lane that provides the traffic.

Stated precisely, given a connection $c = (p, S^c, mx, id)$ and a lane $(s, l) \in S^c \times \mathbb{Z}^s$ on a connection segment, the following requirements correspond to the two consistency requirements.

$$\left.\begin{array}{l} i > 0 \ \wedge \ p_e = p \\ i < 0 \ \wedge \ p_s = p \end{array}\right\} \Rightarrow \exists (s', j) \in S^c \times \mathbb{Z}^{s'} (((s, i), (s', j)) \in mx) \quad (1)$$

$$\left.\begin{array}{l} i < 0 \ \wedge \ p_e = p \\ i > 0 \ \wedge \ p_s = p \end{array}\right\} \Rightarrow \exists (s', j) \in S^c \times \mathbb{Z}^{s'} (((s', j), (s, i)) \in mx) \quad (2)$$

Here, the left sides of the equations accommodate the two cases of segment and connection alignment, namely the cases where either the start or the end point of the segment is located at the connection.

Similarly, we require that (3) movement coming from a lane that starts at a connection is prohibited, and that (4) movement to a lane that ends at a connection is prohibited.

Example 4 Figure 5 depicts a fragment of a road network. Here s_1, s_2, and s_3 are segments of a connection $c = (p, S^c, mx, id)$. Bold and non-bold arrows indicate lane configurations and allowed movements over the connection, respectively. Figure 5a illustrates Eqs. 1 and 2. Here, segment s_3 has two opposite lanes. Thus, there must be at least one lane that receives traffic and one lane that provides traffic over the connection for the lanes on segment s_3. In the figure, $\{((s_3, -1), (s_1, 1)), ((s_2, -1), (s_3, 1))\} \subseteq mx$. Figure 5b illustrates the third constraint.

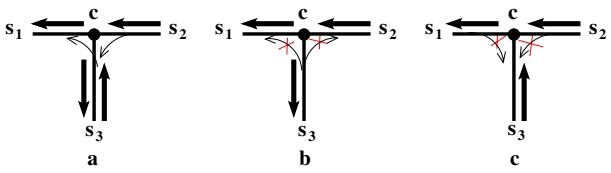


Fig. 5 Constraints imposed by traffic regulations. **a** Movement over the connection that must be present (thin lines). **b** and **c** Movements over a connection that must not be present

Here, s_3 has one lane, which starts at the connection. Thus, there should be no allowed movement from $(s_3, 1)$ to lanes on s_1 or s_2. Figure 5c illustrates the similar case for the fourth constraint.

Apart from the constraints that maintain the consistency of the model, the values of a connection matrix are determined by the traffic regulations that apply to the part of a road represented by the connection. A matrix may reflect regulations such as "left turn not allowed," "movement straight and right only", and "do not enter."

4.2 Storage efficiency

We note that efficient storage has not been a main consideration in the design of the 2D representation. Rather, the objective has been to devise a rich representation of transportation networks at the lane level that may be used as a basis for demonstrating how a complex road network may be represented by a graph model. Consequently, a more storage-efficient representation can be envisioned. For example, linear referencing techniques may be applied to the representation to achieve higher storage efficiency [11].

We also note that computing and storage technologies advance rapidly. We may well expect the vehicles of the future to be equipped with powerful computers with ample storage available. (Current navigation systems already come with tens of gigabytes.)

4.3 Data and query points

Location-based queries in a road network setting usually concern two types of objects: moving objects (query points that capture mobile users), and static (data points that capture stationary objects of interest) objects. For a query point, the key characteristic that we capture is the most recent know position, i.e., the most recent position sample, in the road network. Descriptive properties, as those captured for static points, covered next, can also be introduced.

For stationary objects, we capture the points of their accesses to the network and also descriptive properties. The points of access are positions within the network, through which the objects are accessed (e.g., an entryway into a parking area). The descriptive properties classify the objects with respect to search criteria. A more formal definition follows.

A *data point* is an object of interest. It is characterized by a set of non-spatial properties and a set of locations as well as an ID. For a data point $dp = (prop, loc, id)$, element *prop* is a set of properties that describe the data point. These properties may be used in selection predicates of queries to identify the type or other characteristics

of an object. As our focus is on spatial aspects, we do not elaborate on these, but rather leave *prop* as a placeholder for non-spatial properties.

Next, *loc* is the set of locations that may be associated with an object. Each location *lc* in the set is a four-tuple (p, s, acc), where element p is a point and s is a segment such that p is located on s. Together, these two elements provide the road network coordinates of the location. The third element *acc* indicates the lanes from which the object is *accessible* at the point of the segment, specifically $acc \subset \mathbb{Z}^s$, i.e., it is a subset of all lane numbers for the segment.

Last, the *id* of an object makes it possible to distinguish possibly multiple objects with the same properties and at the same road network location(s). Multiple fast-food outlets that are accessible through the same entrance into a parking area can be an example.

The location of an object is determined by two factors: the location point's geographical location with respect to the location of the segment and by traffic regulations imposed at the location point. This design reflects real-world road settings in that a single object may be accessible from several road segments and from specific lanes along them, e.g., a gas station at an intersection may be accessible from two roads and only from the lane closest to the station of each road. Further, a point of access may have traffic regulations that differ from those of the segment with the point of access. For example, a shopping mall may have multiple entrances where, at some entrances, it is allowed to access the mall by making a left turn, though change of lanes in the direction of the left turn is not allowed at the point of access on a corresponding segment.

Example 5 Consider the example in Fig. 4. The object marked by the pharmacy sign has the property "open." It has two entrances: one from BZ and one from BL. The entrance on BL is from the only lane of the segment. However, a prohibited left turn at the access point on BZ renders the object accessible only from one of two lanes on the segment, i.e., the pharmacy can be described by the tuple ({ "*pharmacy*", "*open*"}, {((4716, 1725), BZ, 1), ((4940, 1870), BL, 1)}, dp#000001).

Finally, *query points* represent the objects that issue queries. A query point $qp = (p, s, lane)$ is given by a point p located on a lane *lane* that belongs to a segment s.

4.4 Road and travel distances

In abstract terms, the movement of an object in a road network, termed a *trajectory*, is given by a continuous curve in the space spanned by the geographical dimensions and the time dimension. The projection of a trajectory onto the geographical space is termed a *path*. During its movement, an object can move between segments, over connections, and can change lanes. We will use the term *stretch* for a part of an object's path where the object stays in the same lane and also stays on the same segment and does not move onto a connection. We can then represent any path as a sequence of stretches.

A stretch is identified by a segment, a lane number, and a pair of delimiting points. Specifically, a stretch is captured by a tuple $str = (p_s, p_e, s, lane)$, where $p_s, p_e \in s$ specify, respectively, the start and an end of the stretch on the lane *lane*

along segment s. The start and the end of a stretch implies the allowed direction of movement on the stretch.

Next, a path *pth* is defined as a list of adjacent stretches $[str_1, \ldots, str_n]$ that satisfies two properties: first, the path is continuous, i.e., the end point of stretch str_i coincides with the start point of the subsequent stretch str_{i+1}; second, the path is *traversable*, i.e., each lane in the path allows traffic in the direction in which it is used in the path and each connection from one lane to another in the path allows movement from the one lane to the other. The set of all paths in a road network is denoted *Pth*.

The *road distance* of traversing a path $pth = [str_1, \ldots, str_n]$ is the sum of the Euclidean distances between start and end points of the stretches in the path list:

$$RD(pth) = \sum_{str_i \in pth} d(p_s^i, p_e^i) \tag{3}$$

where $str_i = (p_s^i, p_e^i, s^i, lane^i)$.

Given a stretch $str = (p_s, p_e, s, lane)$ over segment $s = (p_s, p_e, l, prop)$ and a set of property values $Val^{str} = \{value | (property, value, lanes) \in prop \ \wedge \ lane \in lanes\}$ that affect the movement along the lane of the stretch, the *travel distance* of the stretch is defined as the Euclidean distance between the start and end points, modified by the properties of the lane:

$$TD^{str}(p_s, p_e) = d(p_s, p_e) \cdot \prod_{val \in Val^{str}} val \tag{4}$$

Given a path $pth = [str_1, \ldots, str_n]$, the *travel distance* of the path is the sum of the travel distances of all stretches in the path:

$$TD(pth) = \sum_{str_i \in pth} TD^{str}(p_s^i, p_e^i) \tag{5}$$

where $str_i = (p_s^i, p_e^i, lane^i)$.

Building on these concepts, we define the *road* and *travel distances from point* p_a *to point* p_b (or a distance between the two points) in a road network as the minimal road and travel distances of a path between the two points. Formally, let Pth^{ab} denote the set of all paths from p_a to p_b. Then the road distance from p_a to p_b is $RD(p_a, p_b) = \min_{pth \in Pth^{ab}}(RD(pth))$, and the corresponding travel distance is $TD(p_a, p_b) = \min_{pth \in Pth^{ab}}(TD(pth))$.

5 Graph representation

This section addresses first the modeling of a road network and then the modeling of data and query points for the more abstract graph representation.

5.1 Road networks

The graph representation of a road network is defined as a three-tuple $RN^G = (G, co\mathcal{E}, ch\mathcal{E})$, where G is a directed, labeled graph with loops and multiple edges, $co\mathcal{E}$ is a binary, so-called co-edge, relationship on edges, and $ch\mathcal{E}$ is a binary relationship on edges, termed a change-edge relationship.

Graph $G = (V, E)$ is a two-tuple, where V is a set of *vertices* and E is a set of *edges*. An edge e is a five-tuple $e = (v_s, v_e, w, l, id)$, where $v_s, v_e \in V$ is the *start* and *end vertex* of the edge, respectively. Edge e can be traversed from the v_s to the v_e only. Element w is the *weight* of the edge and represents the travel distance of the part of the road network represented by the edge. Element l is the *length* of the edge, capturing the actual length (or road distance) of the part of the road network represented by the edge. Element *id* identifies the edge among other edges that have the same start and end vertices.

As for the 2D representation, it is required that each vertex v has at least one edge that ends at v (denoted an *incoming edge* of vertex v) and at least one edge that starts at v (denoted an *outgoing edge* of vertex v). Thus, $\forall v \in V\ (\exists e_1, e_2 \in E\ ((v = v_s^1) \wedge (v = v_e^2)))$, where $e_i = (v_s^i, v_e^i, w^i, l^i, id^i)$ for $i = 1, 2$.

A traversal through a graph normally starts and ends at a vertex, and it covers a sequence of edges. However, because it is at times allowed for a vehicle moving in a road network to make a u-turn, or to change lanes (and edges that correspond to lanes), this simple scheme must be extended as shown next.

The co-edge relation $co\mathcal{E} \subseteq E \times E$ captures pairs of edges that represent pairs of lanes for which it is allowed to make a u-turn from the first lane of the tuple to the second lane of the tuple. A u-turn can be made only to the opposing traffic lane that is closest to the current traffic direction. Intuitively, a moving object may jump from one edge in a co-edge relationship to the other edge in the relationship without having reached a vertex. A moving object may overtake a car in front of it by temporarily entering an oppositely directed lane if the pair consisting of the initial lane and the oppositely directed lane belongs to the co-edge relation.

The change-edge relation $ch\mathcal{E} \subseteq E \times E$ captures pairs of edges that represent pairs of lanes where it is possible to change from the first lane in the pair to the second. Lane changes are restricted to lanes in the same traffic direction. The intuition behind the relation is the same as for u-turns—that of a moving object jumping from the first lane to the second without visiting a vertex.

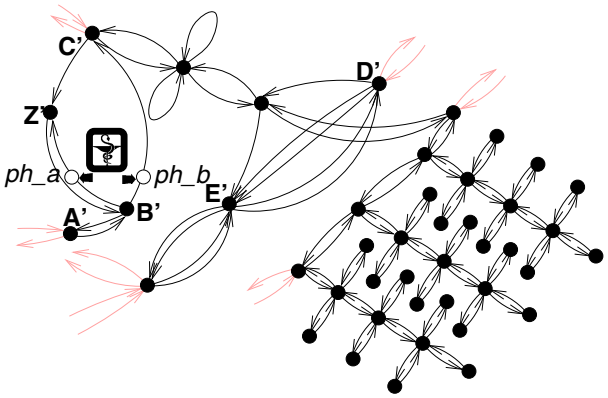
The co-edge and change-edge relations are kept separate due to their different semantics and their different usages during querying. We offer more detail on the two relationships in Section 6.

Example 6 Consider the graph representation of a road network depicted in Fig. 6. Example edges corresponding to the previously presented 2D representation are denoted as follows: $\mathrm{B'Z'} = (\mathrm{B'}, \mathrm{Z'}, 538, 326, \mathrm{e00000001})$, $\mathrm{Z'B'} = (\mathrm{Z'}, \mathrm{B'}, 489, 326, \mathrm{e00000002})$, and $\mathrm{B'C'} = (\mathrm{B'}, \mathrm{C'}, 2560, 1280, \mathrm{e00000003})$. The edge $\mathrm{B'Z'}$ is a co-edge of the edge $\mathrm{Z'B'}$ and vice-versa, i.e., $\{(\mathrm{B'Z'}, \mathrm{Z'B'}), (\mathrm{Z'B'}, \mathrm{B'Z'})\} \subseteq co\mathcal{E}$. Finally, edge $\mathrm{D'E'}_1 = (\mathrm{D'}, \mathrm{E'}, 650, 500, \mathrm{e00000004})$ and edge $\mathrm{D'E'}_2 = (\mathrm{D'}, \mathrm{E'}, 650, 500, \mathrm{e00000005})$ form a pair of change-edges. Assuming that the allowed change is mutual, set $change\mathcal{E}$ contains a tuple for each of the allowed changes.

5.2 Data points

Queries concerning data points issued against the graph representation are typically interested in *where* a data point is located, rather than in whether it merely exists. Thus, the essential characteristic that describes a data point in graph is its location. A location is captured by preserving the semantics of the 2D representation, i.e., it is

Fig. 6 Graph representation



captured as a point in the graph. The location of a point is expressed in terms of graph weight and Euclidean distance with respect to the start of an edge. The two measures allow queries that concern Euclidean, or weighted, or both locations to immediately determine these.

Additional characteristics of graph data points can be extracted using the association with the 2D representation of the data point. The 2D representation includes descriptive attributes that allow us to distinguish among different types of data points, and it identifies the geographical location of a data point.

To be more specific, the set of data points in a graph G is denoted by DP^G.

A data point $dp = (e, pos_w, pos_l, id)$ is a four-tuple, where $e = (v_s, v_e, w, l, id)$ is the edge on which dp is located and elements pos_w and pos_l capture the *weight* and *length* (travel and road distance) of the path corresponding to the section of the road in-between the locations of v_s and dp. Notice that $pos_w \in [0; w]$ and $pos_l \in [0; l]$, i.e., the data point cannot be located outside edge e. Finally, the id references the data point in the 2D representation.

Example 7 Our example (Fig. 6) includes two graph representations of a single data point that represents a pharmacy: $ph\_a = (\mathrm{B'Z'}, 330, 200, \mathrm{dp\#000001})$ and $ph\_b = (\mathrm{B'C'}, 438, 219, \mathrm{dp\#000001})$.

5.3 Query point

A query point captures an object the location of which is capable of changing continuously, and this location is the essential property of the object. The location is essential for tracking, for providing real-time guidance, for identifying relevant, time-dependent conditions, e.g., for identifying opposing traffic of moving objects.

Modeling of object locations at high fidelity requires knowledge of the location continually as it changes. This, typically, entails location prediction, i.e., extrapolation from sampled locations provided by a positioning devices and, possibly, from statistical data. Put differently, the location has to be known from now on and into the near future.

A query point is captured by a triple: $qp = (e, pos_w, pos_l)$ where the elements of the triple correspond to those used for data points (see the previous section) and capture the location of the query point.

Several comprehensive solutions exist for the modeling and tracking of mobile objects [5], [6], [27], [31]. Coverage of these is beyond the scope of this paper.

6 Transformation

This section presents a mapping from the 2D model of a road network to the graph model of a road network. This mapping constructively demonstrates how it is possible to obtain an augmented graph representation from an existing 2D representation.

We present the transformation as a set of mappings, each of which maps a different part of a 2D representation to graph constructs. Transformations for road networks, data points, and query points are provided. The combination of the mappings provides a single, complete transformation $t : RN^{2D} \rightarrow RN^{G}$ of any 2D representation to a corresponding graph representation.

6.1 Road network transformation

The road network transformation maps sets of 2D segments and connections to sets of graph edges, vertices, change-edges, and co-edges. The general idea of the transformation is to identify sequences of lane stretches, called chains, that can be treated as single edges. Connections at the ends of chains are transformed into vertices.

6.1.1 Chains

We define the concept of a set of chains that corresponds to a road network as a set of paths with specific individual properties, where the paths cover the entire road network, do not overlap, and are maximal. A path, $pth = [str_1, \ldots, str_n]$, is a *chain* if it satisfies the following constraints:

1. Every stretch *str* of the path spans an entire segment. Specifically, the start and end point of every stretch, denoted p_s^{str} and p_e^{str}, respectively, has to correspond to a delimiting point of the segment on that the stretch lies. Such a segment is called a *corresponding segment* of stretch *str* and is denoted s^{str}.
2. All stretches in *pth* must lie on the same lane. Thus, with $lane^i$ and $lane^j$ belonging to str^i and str^j, respectively, we have: $\forall str_i, str_j \in pth\ (|lane^i| = |lane^j|)$.
3. Adjacent stretches must have a corresponding segment each so that these segments meet. Stated more precisely, adjacent stretches $str_i, str_{i+1} \in pth$ must have corresponding segments s^{str_i} and $s^{str_{i+1}}$ such that a delimiting point of s^{str_i} and a delimiting point of $s^{str_{i+1}}$ coincide at some connection. In addition, no other segments must exist that start or end at this connection.
4. The connection between segment s^{str_i} and segment $s^{str_{i+1}}$ corresponding to adjacent stretches str_i, str_{i+1} must allow movement corresponding to the lane-change and u-turn restrictions on s^{str_i} and $s^{str_{i+1}}$. For every pair of lanes $(k, l) \in v$ of segment s^{str_i} such that the movement along lane k is towards the connection, and k and l are both positive or both negative, connection matrix *mx* must contain entry $((s^{str_i}, k), (s^{str_{i+1}}, l))$. This case ensures that changes between lanes in the same direction allowed on s^{str_i} are also allowed at the connection. Next, if $k, l \in$

$\{-1, 1\}$, i.e., if k, l are of different signs, mx must contain entry $((s^{str_i}, k), (s^{str_i}, l))$. This case ensures that if a u-turn is allowed on s^{str_i}, a u-turn is also allowed at the connection. Finally, corresponding constraints apply to segment $s^{str_{i+1}}$.

5. Segments corresponding to adjacent stretches $str_i, str_{i+1} \in pth$ (connected by a connection), must have identical property sets (meaning the same lane configuration and the same lane properties), i.e., *prop* components of the corresponding segments must be equal.
6. Path list must not contain repetitive stretches, i.e., $\forall str_i, str_j \in ch (i \neq j \Rightarrow str_i \neq str_j)$.

The first condition ensures that a path is at least as long as some segment, it identifies building blocks of the path. The second condition ensures that the path stays on the same lane over all stretches of the path. The third and the fourth conditions require that there is no lane configuration change in the path (required for transformation of segments into edges). The fifth condition ensures that the lane in a path can be given the same weight per length unit in the graph representation. The sixth condition eliminates the possibility of loops.

Example 8 Consider Fig. 4. Paths that satisfy the chain constraints may be delimited by: (1) intersections of more than two roads, e.g., by the intersection at point C; (2) by the end of a road, e.g., as depicted in right bottom part of the figure; and (3) changes in road properties, e.g., at a connection delimiting stretches with different speed limit on a lane, or a different lane configuration, as it happens at point Z (see Fig. 1 for an illustration). In the latter situation, point Z delimits three paths: a path pth_{CZ} that covers a road with a single stretch, and two oppositely directed paths pth_{BZ}, pth_{ZB} that cover a road with two stretches in opposite directions (induced by two lanes in opposite directions).

The set of chains describing a road network must posses two properties: each lane of a segment in the road network is covered by exactly one chain, and no two chains can be combined into one chain. For example, it is possible for a one-lane road with the same properties through its entirety to be captured by multiple stretches. It is then also possible for this road to be covered by different sets of chains. One such set may contain a single chain for all stretches, while another may contain one chain for each stretch. In this case, the last requirement above implies that the singleton set is used.

For the chain $ch = [str_1, \ldots, str_n]$, we will use the term *start* and *end* connection, denoted c_s and c_e, respectively, for the connection co-located with the start point of str_1 and end point of str_n.

6.1.2 Connections to vertices

We map 2D connections to graph vertices and to so-called zero-edges (the set of such edges is given by $E^{C0} \subseteq E$). The idea is to transform every 2D connection delimiting a chain into a vertex if there are no movement restrictions over the connection. In the case where movement restrictions are present at the connection, the connection is represented by a structure of vertices and edges that preserve the movement restrictions of the connection in the graph representation, without altering the results of distance-related calculations in the graph.

Before we detail the mapping, we introduce the notions of *incoming* and *outgoing* segments for a connection c, denoted S^c_{in} and S^c_{out}, respectively. These are the segments that have lanes allowing movement to (referred to as *incoming lanes*) and from (referred to as *outgoing lanes*) the connection, respectively. Stated precisely,

$$S^c_{in} = \{s \in S^c \mid \exists l_s \in \mathbb{Z}^s \ (\ \exists (s', l_{s'}) \in S^c \times \mathbb{Z}^{s'} \ (((s, l_s), (s', l_{s'})) \in mx))\}$$

$$S^c_{out} = \{s \in S^c \mid \exists l_s \in \mathbb{Z}^s \ (\ \exists (s', l_{s'}) \in S^c \times \mathbb{Z}^{s'} \ (((s', l_{s'}), (s, l_s)) \in mx))\}$$

The first formula captures the segments s that have at least one lane l_s *from* which it is allowed to move over the connection to a lane $l_{s'}$ of a segment s'. The second captures the segments s that have at least one lane l_s *onto* which movement is allowed over the connection from a lane $l_{s'}$ of a segment s'. Note, that the two sets may overlap and that their union is equal to S^c.

Example 9 With the movement directions on segments s_1, s_2, and s_3 given by the arrows in Fig. 7b, the sets of incoming and outgoing segments for connection c are $S^c_{in} = \{s_1, s_2\}$ and $S^c_{out} = \{s_1, s_2, s_3\}$.

With these definition in place, we first map a connection to a single vertex if the connection allows movement from each of its incoming lanes to each of its outgoing lanes. The set of all such connections is denoted C^s and is formally defined as follows:

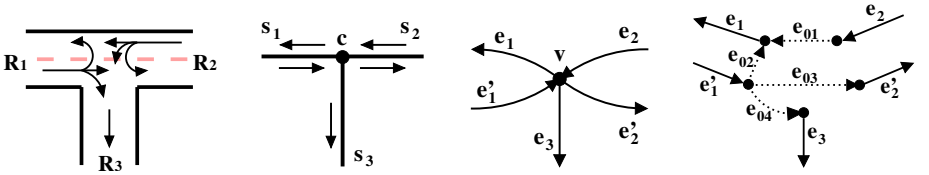
$$C^s = \Big\{c \in C \mid c = (p, S^c, mx, id) \wedge \forall\ ((s_{in}, l_{s_{in}}), (s_{out}, l_{s_{out}})) \in (S^{c_{in}} \times \mathbb{Z}^{s_{in}}) \times (S^{c_{out}} \times \mathbb{Z}^{s_{out}})\ (((s_{in}, l_{s_{in}}), (s_{out}, l_{s_{out}})) \in mx\ )\ \wedge\ \mathcal{P}(c)\Big\},$$

where predicate $\mathcal{P}(c)$ ensures that connection c is the start or end connection of some chain. Specifically, $\mathcal{P}(c) = (\exists ch \in Ch(c = c_s \vee c = c_e))$, where c_s and c_e is start and end connection of chain ch.

The formula requires that each possible pair of an incoming lane and an outgoing lane of the connection belong to the connection matrix mx, i.e., that mx allows movement from the former lane to the latter lane. The transformation of connections into single vertices is thus a one-to-one mapping that creates one vertex in V^{C^s} for each connection in C^s:

$$t^{C^s} : C^s \rightarrow V^{C^s}$$

where $V^{C^s} \subseteq V$.



a The Road Map **b** 2D Representation **c** Graph Representation (Without Restrictions) **d** Graph Representation (With Restrictions)

Fig. 7 Representations of an intersection

Example 10 Considering again Fig. 7, connection c is transformed into a vertex v as presented in Fig. 7c, if and only if movement is allowed from every incoming lane of the connection to every outgoing lane, i.e., if no restrictions are posed on the movement over the intersection. Specifically, given the allowed movement depicted in Fig. 7a and assuming that c is the start connection of s_1, s_2, and s_3, connection c belongs to C^s iff the connection matrix mx contains six tuples corresponding to the six arrows indicating the allowed movement over the connection, i.e., $mx = \{((s_1, -1), (s_1, 1)), ((s_1, -1), (s_2, 1)), ((s_1, -1), (s_3, 1)), ((s_2, -1), (s_2, 1)), ((s_2, -1), (s_1, 1)), ((s_2, -1), (s_3, 1))\}$.

Next, a connection is transformed into multiple vertices if the movement from at least one incoming lane to at least one outgoing lane is not allowed. These connections are given as follows:

$$C^m = \big\{c \in C \mid c = (p, S^c, mx, id) \wedge \exists\, ((s_{in}, l_{s_{in}}),\ (s_{out}, l_{s_{out}})) \in (S^c_{in} \times \mathbb{Z}^{s_{in}}) \times (S^c_{out} \times \mathbb{Z}^{s_{out}})\ (((s_{in}, l_{s_{in}}), (s_{out}, l_{s_{out}})) \notin mx) \wedge \mathcal{P}(c))\big\}.$$

The formula requires a connection to have an incoming lane and an outgoing lane such that the pair does not belong to the mx of the connection, i.e., such that mx does not allow movement from the former to the latter.

The goal of the transformation of a connection into multiple vertices is to precisely capture the movement allowed on the connection. This is achieved by modeling every lane of every connection segment as a separate vertex, and by introducing additional edges, one for each allowed movement direction over the connection.

Multiple vertices for a connection are generated by a mapping that generates one vertex for each (*connection, segment, lane*) triple. Here *segment* and *lane* identify a lane, while *connection* specifies either the start or the end of the lane. Thus, vertices for lanes are generated by a one-to-one mapping that generates one vertex from every lane of every connection segment for a given connection:

$$\nu : C^m \times S^c \times \mathbb{Z}^s \to V^{C_m},$$

where $V^{C_m} \subset V$.

Vertices that represent the same connection are connected by edges of zero length and weight, called *zero-edges*. Specifically, zero-edges in the graph form the following set:

$$E^{C0} = \{(v_s, v_e, 0, 0, id) \in E \mid \exists c \in C^m\ (c = (p, S^c, mx, id) \wedge (\exists (s, l_s), (s', l_{s'}) \in S^c \times \mathbb{Z}^s\ (((s, l_s), (s', l_{s'})) \in mx\ \wedge v_s = \nu(c, s, l_s)\ \wedge\ v_e = \nu(c, s', l_{s'}))))\}.$$

The definition requires that the start and end vertices of a zero-edge are mapped using the function ν from an incoming and an outgoing lane of the same connection, respectively. This connection must allow movement from the incoming to the outgoing lane.

Then the transformation of a connection c into multiple vertices produces one vertex for each lane of incoming and outgoing connection segments of c, and into a set of corresponding zero-edges:

$$t^{C^m} : C^m \to 2^{V^{C^m}} \times 2^{E^{C0}}$$

The sets C^s and C^m comprise all transformation-relevant connections—the remaining connections are inside chains and are captured "within" edges. The set of all vertices V is the union of the two sets of vertices V^{C^m} and V^{C^s}. In conclusion, the connections in C are transformed to the graph representation by mappings t^{C^s} and t^{C^m}.

Example 11 Assume that left turn and u-turn over the connection c are not allowed for vehicles coming from segment s_2 (see Fig. 7). This requires transformation of the connection into multiple vertices and zero-edges. Vertices created for each incoming and outgoing lane, i.e., for each triple $(c, s_1, -1)$, $(c, s_1, 1)$, $(c, s_2, -1)$, $(c, s_2, 1)$, $(c, s_3, 1)$, form a set $V_i^{C^m} \subset V^{C^m}$ with five vertices.

Next, the vertices are connected by zero-edges, as depicted in Fig. 7d. Specifically, edges e_{02}, e_{03}, and e_{04} connect the vertex of the incoming lane of s_1 to the vertices of the outgoing lanes of s_1, s_2, and s_3. Further, the single zero-edge e_{01} connecting the vertex of the incoming lane of s_2 with the vertex of the outgoing lane of s_1 effectively captures the prohibited left turn and u-turn. The four zero-edges introduced form the set $E_i^{C0} \subset E^{C0}$.

6.1.3 Segments to edges

We proceed to map lanes of segments to edges and to create appropriate co-edge and change-edge relationships. We also maintain an auxiliary data structure the maps each edge to the lane it represents. This mapping is needed during graph refinement (as presented in Section 7) and is subsequently discarded. If no refinement is intended, the mapping can be disregarded.

The idea is to represent each chain by a single edge. Additionally, pairs of edges are registered as co-edges if the corresponding pairs of chains allow u-turn from the lane corresponding to the first chain to the lane corresponding to the second chain. Analogously, pairs of edges are registered as change-edges if the lanes of the corresponding pairs of chains allow lane change from the lane corresponding to the first chain to the lane corresponding to the second chain.

The transformation of chains is a one-to-one mapping: $t^{ch} : Ch \rightarrow E^{ch}$, where $E^{ch} \subseteq E$. The transformation maps each chain ch to an edge $e = (v_s, v_e, w, l, id)$ as follows:

- v_s, v_e derive from the mapping of the start and end connections c_s and c_e of chain ch. The mapping differs for connections mapped to single vertices versus connections mapped to multiple vertices. The mapping particular to each connection is accomplished by a function $f : C \times Ch \rightarrow V$ that identifies a vertex corresponding to a given connection $c = (p, S^c, mx, id)$ and chain ch:

$$f(c, ch) = \begin{cases} t^{C^s}(c) & \text{if } c \in C^s \\ v(c,\ f'(c, ch)) & \text{if } c \in C^m \end{cases} \tag{6}$$

  where $f'(c, ch) = \{(s, l_s) | s \in S^c \wedge l_s \in \mathbb{Z}^s \wedge l_s \in ch\}$; note that $card(f'(c, ch)) = 1$. Then we let $v_s = f(c_s, ch)$, and $v_e = f(c_e, ch)$.
- w is equal to the travel distance of the chain ch, i.e., $w = TD(ch)$.
- l is equal to the road distance of the chain ch, i.e., $l = RD(ch)$.
- id is a unique identifier.

Each edge $e \in E^{ch}$ is also mapped to the lane number of the stretches in the chain from which the edge was transformed.

Let e and e' be the edge generated for chain ch and ch', respectively. An element (e, e') is inserted into the $co\mathcal{E}$ relation if ch' captures a lane accessible from the lane captured by ch, and if the stretches in ch are aligned in the opposite direction than stretches in ch'. Next, an element (e, e') is inserted into the $ch\mathcal{E}$ relation if chain ch' captures a lane accessible from the lane captured by ch and edges e and e' allow movement in the same direction.

Example 12 The left part of Fig. 8 depicts a fragment of the road network shown in Fig. 4, with the addition of bigger circles that mark the delimiting points of chains. Assume that the depicted segments have the properties presented in Table 1. Then, the example captures chains ch_{CZ}, ch_{BC}, ch_{BZ}, and ch_{ZB}. Moreover, assume that the connection at point B does not allow movement from the segment ZB to the segment BL. Then the chain $ch_{\mathrm{BC}} = [str_{\mathrm{BL}}, str_{\mathrm{LN}}, str_{\mathrm{NM}}, str_{\mathrm{MC}}]$ (here, e.g., str_{BL} denotes the stretch from B to L on the only lane of the particular road) is transformed into an edge $\mathrm{B}'_3\mathrm{C}' = (v_s, v_e, w, l, id)$ where $v_s = \nu(\mathrm{B}, f'(\mathrm{B}, ch_{\mathrm{BC}})) = \mathrm{B}'_3$, $v_e = t^{Cs}(\mathrm{C}) = \mathrm{C}'$, $w = TD(ch)$, $l = RD(ch)$, and $id = \mathrm{e}00000003$). Similarly, the chains ch_{BZ} and ch_{ZB} are transformed into edges $\mathrm{B}'_2\mathrm{Z}'$ and $\mathrm{Z}'\mathrm{B}'_1$. Additionally, $(\mathrm{B}'_2\mathrm{Z}', \mathrm{Z}'\mathrm{B}'_1)$ and $(\mathrm{Z}'\mathrm{B}'_1, \mathrm{B}'_2\mathrm{Z}')$ are inserted into the $co\mathcal{E}$ relation to indicate that u-turn is allowed in both directions.

6.2 Data point transformation

The next step is to transform 2D data points into graph data points. The idea is to map every *location* of a data point in 2D into a separate data point in the graph. Because a 2D data point can have several locations and can be accessible from several lanes at every location, a single 2D data point may have several corresponding data points in the graph.

First, recall that a data point in the 2D representation is a tuple $dp = (prop, loc, id)$. Every location $lc = (p, s, acc)$ in the set loc of the tuple can be accessed from multiple lanes; thus, a single location can produce a set of graph data points. This is handled by the mapping $t^{Loc} : DP^{2D} \rightarrow 2^{DP^G}$. Specifically, a 2D location and a lane $l \in acc$ is mapped to a graph data point $dp = (e, pos_w, pos_l, id)$ as follows:

– Edge e corresponds to the chain ch capturing a lane l from which the object is accessible. (Thus, $e = t^{Ch}(ch)$.)

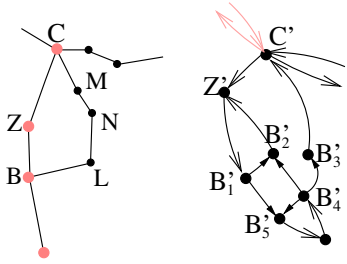


Fig. 8 Chains of segments transformed into edges

- The position according to length, pos_l, is defined based on the chain ch that corresponds to the edge e. The position is the road distance of traversing the part of ch from its start to the location of the data point. Let $ch = [str_1, \ldots, str_k, \ldots, str_n]$, such that str_k starts at point p_s^k and stretches over segment s, additionally let $ch' = [str_1, \ldots, str_{k-1}]$. Then $pos_l = RD(ch') + d(p_s^k, p)$.
- The position according to weight, pos_w, is the sum of the travel distance of path ch' (defined above) and the travel distance of the segment's residual, i.e., $pos_w = TD(ch') + TD^{str}(p_s^k, p)$.
- Element id refers to the id of the 2D data point.

Example 13 Consider the pharmacy located close to two roads in Example 5. The pharmacy is described by a tuple ({"*pharmacy*" , "*open*"}, {((4716, 1725), BZ, 1), ((4940, 1870), BL, 1)}, dp#000001). It thus has two locations, each of which has one lane. The element id references the 2D data point. The resulting graph representation of the pharmacy consists of $ph_a = (\mathrm{B'Z'}, 330, 200, \mathrm{dp\#000001})$ and $ph_b = (\mathrm{B'C'}, 438, 219, \mathrm{dp\#000001})$.

6.3 Query point transformation

We conclude the transformation procedure by mapping 2D query points to graph query points: $t^{qp} : qp^{2D} \rightarrow qp^G$. Recall that a 2D query point is captured by a sampled position: a point, a segment, and a lane, i.e., $qp^{2D} = (p, s, lane)$. A graph query point is modeled by a weight and length along an edge, i.e., $qp^G = (e, pos_w, pos_l)$. The edge e, the position according to length pos_l, and the position according to weight pos_w are defined similarly to the corresponding definitions for graph data points.

7 Refinement of the graph representation

Our graph data structure includes two non-traditional aspects: it allows multiple edges between pairs of vertices, and it captures so-called lateral connectivities between edges. The objective of refinement is to simplify the graph representation of a road network so that it differs only little from that obtained when using a classical directed, weighted graph. In addition, the refinement aims to minimize the size of a graph. The simplifications possible depend on the intended uses of a graph. We say that simplifications must preserve the use-specific semantics associated with a graph. We require to preserve lanes in the reminder of the section.

7.1 Use-imposed limitations on refinement

We proceed to examine use scenarios for the graph representation to obtain insight into the semantics that the representation carries for these uses. Our target uses include routing, transportation simulation, and tracking.

In *route search*, also called routing, the objective is to determine a route in a graph from one graph location to another. The route is a collection of edges and lateral movements between them that yield a traversal from the source to the destination. The shortest traversals are often of particular interest. This use poses a requirement

for the accurate capture of lane information in the graph representation, as different lanes in a traversal imply different routes.

Next, in *transportation simulation*, the movements of objects in a transportation network are simulated. The objects are assumed to adhere to the traffic regulations—to follow lanes and make turns only where they are allowed; and to adhere to simple physical constraints—e.g., to change lanes when overtaking another object instead of moving over/through the object in front. A primary use of simulations is to capture or model object movement in the graph representation in order to perform calculations over their location, e.g., to predict congestion based on the prediction of the future positions of objects, or for verifying hypotheses about transportation networks by realistic simulations. Such realistic simulation needs positioning at the granularity of lanes.

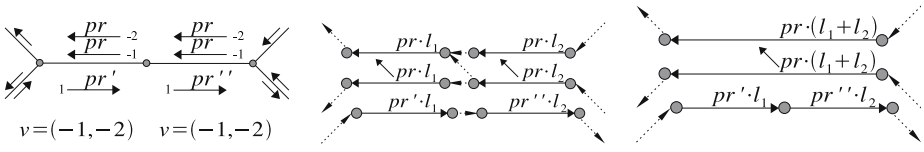
Finally, in *tracking* of moving objects, two positions for each object are of interest: the actual position and the position as believed by a central server. An object positions itself in the 2D representation. The server predicts the position of an object by means of the most recent information received from the object and a prediction strategy; and the object is aware of this prediction. The object then issues an update to the server when the server prediction degrades, thereby meeting an accuracy guarantee on the server side. Tracking can be performed at both the granularity of roads and lanes. Using the latter allows to address a broader range of issues.

In order to reduce the size of a graph, we apply a procedure that iteratively attempts to apply two types of modifications.

7.2 Refinement procedure

A graph element is *superfluous* if the same semantics can be expressed without the element.

Consider a change of a property from pr' to pr'' on some lane of a road. Presented graphically, Fig. 9 schematically captures a part of a road network where, first, a stretch of a two-way road meets a stretch of a one-way road. These two then merge into a single road with three lanes and then they split into two roads. Each traffic lane in the 2D representation (a) is depicted as an arrow with a label. Digits capture the numbers of lanes. Lanes with label pr (lanes $-1, -2$) have the same property set, while labels pr' and pr'' indicate a change of properties on a lane (lane 1). Additionally, it is allowed to make a lane change from lane -1 to lane -2, as indicated by the two lane-change restriction sets v ((a) bottom). The transformation of the 2D model yields a graph (b) that depicts each (part of a) lane as an edge with an annotation that captures the weight of the edge. The figure also uses non-connected arrows to capture the change-edge relationships. The refined graph

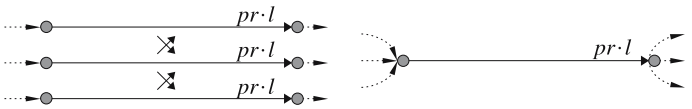


a 2D Indicating Properties of Lanes and Lane Change Restriction

b Corresponding Graph with Change-Edge Relations and Weight Expressions

c Graph With Unnecessary Elements Eliminated

Fig. 9 Transformation example allowing for unnecessary edges and vertices



a Initial Graph **b** Result of Refinement

Fig. 10 Lateral merge of edges

(c) eliminates unnecessary elements: it merges consecutive edges ((c) top) and removes an unnecessary zero-edge ((c) bottom).

Superfluous edges and vertices are removed by a refinement procedure that identifies them, eliminates them, and thus compacts the graph. The result of the refinement is a smaller graph that retains all semantics.

The refinement process starts by scanning the original graph in RN^G for patterns consisting of a subgraph, called *source subgraph*, and of a subset of lateral connectivities. Upon encountering a pattern in the graph, the source subgraph is replaced by a more concise *target subgraph*, and corresponding adjustments are made to lateral connectivities. The process is iterative, i.e., the outcome of a process iteration produces a graph that can possibly be refined.

The naive strategy to multi-graph refinement is to identify as a pattern a subgraph with edges that have the same weight/length ratio and that all have the same pair of start and end vertices, i.e., edges that represent different lanes on the same stretch of a road. The edges in the source subgraph are then merged into one edge (see Fig. 10). This strategy is useful when graphs are used for distance calculation which, generally, tolerates laterally merged lanes. This naive strategy is, however, not acceptable due to our constraint of preserving lane data.

We proceed to consider two refinements: merging of edges that represent the same lane on adjacent stretches of a road (consecutive merges), and an elimination of zero-edges.

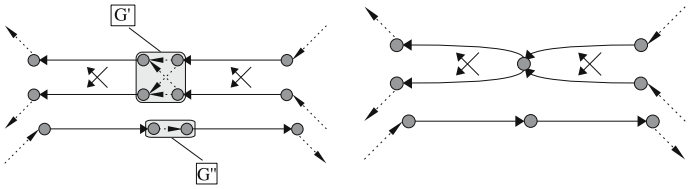
7.2.1 Transformation of zero-edges

The transformation of zero-edges simplifies a graph by replacing a connected subgraph of zero-edges (i.e., the edges with their delimiting vertices) by a subgraph of one vertex. This transformation maps one graph to another

Technically, the idea is to reuse the mechanism that transforms a geographical connection and its connection segments into a single graph vertex. The difference is that the mechanism is now applied to a subgraph of zero-edges (as if it were a connection) rather than to a connection.

The transformation of zero-edges takes a subgraph of zero-edges (subgraphs G' and G'' in Fig. 11) as argument. The vertices in this subgraph function as either *sinks* or *sources*: the end vertices of the zero-edges are the sinks, and the start vertices are the sources. It is required that every sink of the subgraph is connected to every source of the subgraph. Such a subgraph can be mapped into a single vertex. The non-zero-edges that used to start or to end at the sinks and the sources are modified so that they start and end at the single vertex.

The refinement described is most likely to be applicable on roads with few lanes in total, stretches with an increase/decrease in the number of lanes, and laterally separated parts of wider roads.



a Initial Graph with Superfluous Edges **b** Result of Refinement

Fig. 11 Transformation of zero-edge subgraphs

7.2.2 Merging of consecutive edges

The basic idea in this section is to merge two edges with a zero-edge located in-between them into a single edge. This idea is used to define patterns and corresponding transformations called merges of consecutive edges, or a *consecutive merges*.

The source subgraph of the consecutive merge pattern has three parts: a *left subgraph* G^L, a set of *connecting zero-edge(s)* G^0, and a *right subgraph* G^R (note that "left" and "right" are terms used for reference only). The left and the right subgraph, consists of non-zero-edges. Vertices of the connecting zero-edges lie in the left or in the right subgraph. The lateral relationships of the consecutive merge pattern are captured by co-edge co$\mathcal{E}^{LR}$ and change-edge ch$\mathcal{E}^{LR}$ relations between edges in G^L or G^R. Additional constraints differ between patterns (described further).

The refinement constructs a target subgraph by a function that maps the original subgraphs to the target subgraph: $r : G^L \cup G^0 \cup G^R \rightarrow G'$. It also maps from a subset of the co/change-edge relation to the target subset of the relation: $r : 2^{\text{co}\mathcal{E}^{LR}} \rightarrow 2^{\text{co}\mathcal{E}'}$, $r : 2^{\text{ch}\mathcal{E}^{LR}} \rightarrow 2^{\text{ch}\mathcal{E}'}$. The refinement removes all original lateral relationships in co$\mathcal{E}^{LR}$ and ch$\mathcal{E}^{LR}$ from the set co$\mathcal{E}$ and ch$\mathcal{E}$ and, instead, adds the refined relationships in co$\mathcal{E}'$ and ch$\mathcal{E}'$.

The edges in the target subgraph G' are constructed and then added to RN^G. Whenever an edge is created in G' by the refinement (by merging edges in the source subgraph), the edge gets a length and weight that correspond to the sum of the lengths and weights of the edges that it is created from. The edge also gets a unique identifier. The source subgraph edges used for the creation of new edges are discarded. The auxiliary edge-to-lane mapping, described in Section 6.1.3, is also updated with a mapping of the new edge to its lane number (the lane number of the edges that is was created from). This preserves the consistency of the graph representation and allows the created edges to take part in patterns.

Note that when it is possible for a (moving) query point to be located on an edge that is being consolidated during refinement, the refinement mapping must be preserved. This is needed in order to map the query point from its location in the 2D representation to its location on an edge in the refined graph representation. Alternatively, consecutive merge refinement should not be performed. The same line of reasoning applies to data points.

Basic Patterns A constraint common to all basic patterns requires that the graph of the connecting zero-edges is a connected graph.

The patterns and their mappings to targets are presented in terms of subgraphs and lateral relations:

- A *one-way* pattern involves left and right subgraphs that contain edges v_1v_2 and v_3v_4 (we use v_iv_j as a shorthand for an edge that starts at vertex v_i and ends at vertex v_j), such that the edges participate in no lateral relationships. They have equal weight per length unit and it is possible to get from the left edge to the right by a connecting zero-edge v_2v_3—see Fig. 12a.
  The refinement substitutes (by merging) the two edges and the zero-edge with a newly created edge v_1v_4.
- A *two-way single access* pattern involves a left and a right subgraph that each consist of a pair of edges that participate in a single co-edge relationship—see Fig. 12b. The co-edge relation for the left and for the right subgraph consists of tuple (v_3v_4, v_2v_1) and (v_7v_8, v_6v_5), respectively. Connecting zero-edges form a set $\{v_5v_2, v_4v_7, v_4v_2\}$. The first two of these elements indicate the continuity of lanes, and the last element mirrors the co-edge relationship.
  The refinement merges edges v_6v_5 and v_2v_1 that capture parts of the same lane into a single edge v_6v_1. Similarly, edges v_3v_4 and v_7v_8 are mapped to an edge v_3v_8. Tuple (v_3v_8, v_6v_1) captures refined co-edge relationship for the subgraph.
- A *two-way mutual access* pattern is identical to the two-way single access pattern, with a few differences. The differences indicate mutual lateral connectivity: the source left and right subgraphs each participate in additional co-edge relationships (v_2v_1, v_3v_4) and (v_6v_5, v_7v_8)—see Fig. 12c; the additional zero-edge v_5v_7 belongs to the connecting zero-edges; the refined co$\mathcal{E}$ relation contains the additional tuple (v_6v_1, v_3v_8).
- A *one-way single access* pattern, shown in Fig. 12d, is mapped similarly to how the two-way single access pattern is mapped. The differences are due to the two lanes being in the same direction. Thus, edge v_6v_5 and v_2v_1 becomes v_5v_6 and v_1v_2, respectively; zero-edge v_4v_2 becomes v_4v_5; and the co-edge relationships become change-edge relationships. With these adjustments, the mapping for the two-way single access pattern is used.
- A *one-way mutual access* pattern, shown in Fig. 12e, is mapped similarly to how to the two-way mutual access pattern is mapped. As above, differences occur because the two lanes have the same direction. As a result, edge v_6v_5 and v_2v_1 becomes v_5v_6 and v_1v_2, respectively; edges v_1v_2 and v_5v_6 must correspond to the same lane in the road network (as recorded in the edge-to-lane mapping), and edges v_3v_4 and v_7v_8 must also correspond to the same lane in the road

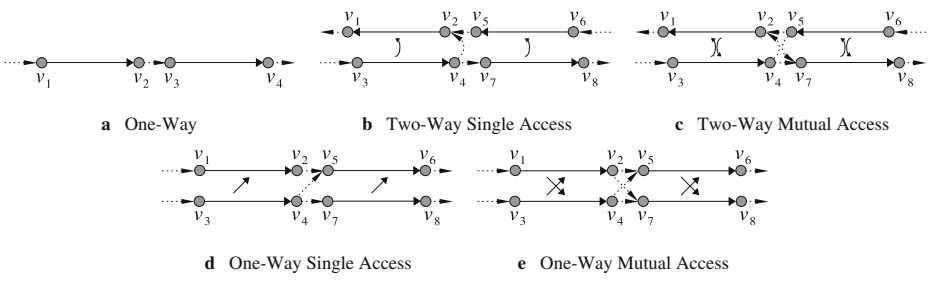


Fig. 12 Patterns for consecutively merge

network; zero-edge v_4v_2 becomes v_4v_5 and zero-edge v_5v_7 becomes v_2v_7; and co-edge relationships become change-edge relationships.

Example 14 Consider the highway exit depicted in Fig. 13. In this example, there are no entrances/exits for the traffic that travels in the opposite direction. In this case, a graph representation of the road may contain the one-way mutual access pattern. The left and the right subgraph G^L and G^R of the pattern contains edges v_4v_3, v_8v_7 and v_2v_1, v_6v_5, respectively. The connecting zero-edges v_3v_2, v_3v_6, v_7v_6, v_7v_2 form a connected graph G^0. Also, the lateral connectivity of the pattern includes change-edge relationships (v_2v_1, v_6v_5), (v_6v_5, v_2v_1), (v_4v_3, v_8v_7), (v_8v_7, c_4v_3). The pattern graph and its change-edge connections are substituted by a new subgraph G' and a reduced set of change-edge relationships as depicted in Fig. 13b.

7.2.3 Composite patterns

The patterns considered in the previous section can be combined to form composite patterns, also called *complexes*, that can also be refined.

A complex can consist of an arbitrary number and of arbitrary types of patterns. It is not practical to enumerate the possible complexes and then present a mapping for each. Instead, we give an iterative procedure for identifying complexes in a graph. Following that, we explain how to reuse the mappings of basic patterns when producing the mapping of a complex obtained via the iterative procedure.

1. Identify a basic pattern where some of the edges of the pattern are laterally connected to edges outside of the pattern and name it the *current pattern*.
2. Identify current pattern edges that are laterally connected to the remainder of the graph and name them *boundary edges*. If no new *boundary edges* are found, skip Steps 3 and 4.
3. For each pair of boundary edges that capture the same lane, determine whether they form a pattern with laterally connected edges that are not part of the current pattern; each such pattern is an *iterative step pattern*.
4. For each iterative step pattern, substitute the current pattern with the union of the current and of the iterative step patterns and goto Step 2.
5. Check that the connections of the *current pattern* to the remainder of the graph do not prevent refinement (as presented in Section 7.2.4). If preventing connections are present, set the current pattern to the empty set.
6. The *current pattern* is empty or contains the identified pattern.

The procedure starts by identifying a basic pattern. It then attempts to grow the pattern by identifying one or two patterns that are "adjacent" to the initial,

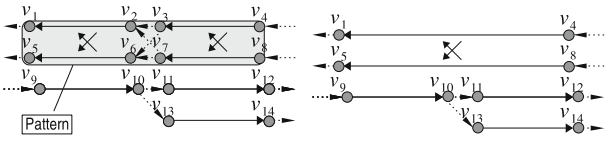


Fig. 13 Refinement of a basic pattern

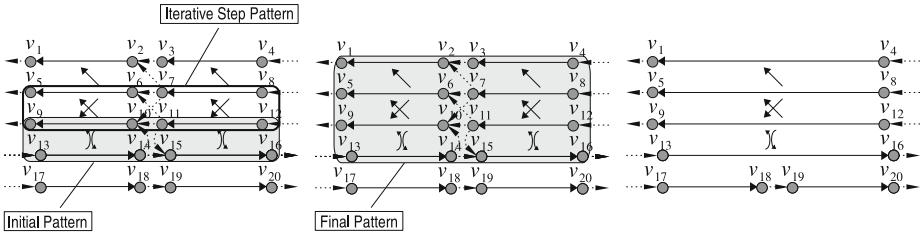
a One-Way Mutual Access Pattern **b** Refined Graph Representation

current pattern and by adding the identified patterns to the current one, i.e., the initial, current pattern is substituted with the pattern that is constructed by including all elements of the current and of the identified patterns. The substituted pattern is grown during further iterations until no additional adjacent patterns can be identified. Next, connections between the substitute pattern and the remainder of the graph are examined to determine whether the substitute pattern is suitable for refinement. This examination is elaborated upon in Section 7.2.4. If the substitute pattern is suitable for refinement, it is identified as a pattern. The identified pattern is either empty, a basic pattern, or a complex.

Example 15 We consider the identification of a complex pattern in the graph shown in Fig. 14a. Assume that the first step has identified the basic two-way mutual access pattern. Two of its edges, $v_{12}v_{11}$ and $v_{10}v_9$, are laterally connected to the two edges v_8v_7 and v_6v_5 that are not in the pattern. We find that the four edges form the basic one-way mutual access pattern. According to the procedure, the two patterns are united into one that is then used as the initial, current pattern in the next iteration. That iteration finds a one-way single access pattern and includes this in a final current pattern composed of edges v_4v_3, v_2v_1, v_8v_7, v_6v_5, $v_{12}v_{11}$, $v_{10}v_9$, $v_{13}v_{14}$, $v_{15}v_{16}$ (see Fig. 14b). Without presenting the examination (to be covered in Section 7.2.4), we state that the final current pattern has no connections that prevent it from refinement and, thus, it is an identified pattern. The identified final pattern is a complex.

The mapping of a complex pattern combines the mappings of the basic patterns that the complex pattern contains. The initial pattern and the patterns of the iterative step are basic patterns. For each of these patterns, we apply the corresponding mapping. In addition, we ensure that no duplicate edges are produced, i.e., we do not allow the same pair of source edges to be mapped to two different edges in different basic pattern mappings.

Example 16 Consider Fig. 14. We refine the complex pattern in Fig. 14b by applying the refinements of basic patterns to the initial pattern and the iterative step patterns. Thus, the two-way mutual access pattern refinement is applied to the initial pattern that is formed of the left and right subgraph G^L and G^R containing $v_{10}v_9$, $v_{13}v_{14}$ and $v_{12}v_{11}$, $v_{15}v_{16}$ respectively, of zero-edge subgraph Z^0 containing $v_{11}v_{10}$, $v_{11}v_{15}$, $v_{14}v_{10}$, $v_{14}v_{15}$, and of four corresponding co-edge relationships. A



a Iterative Step of Identifying a Pattern **b** Maximal Identified Pattern **c** Refined Pattern

Fig. 14 Iterative identification and refinement of a complex pattern

consecutive iterative step pattern is refined according to the one-way mutual access basic pattern refinement. The last refinement is that of the one-way single access pattern. The refinements of the iterative step and of the initial pattern both map edges $v_{12}v_{11}$, $v_{10}v_9$ to a refined edge $v_{12}v_9$. The edge is created by only one of the two mappings.

A complex may have arbitrary size and structure. However, in practical applications to real road infrastructures, we may expect that a single complex groups at most one two-way access pattern and several one-way single/mutual access patterns. Such complexes occur when modeling multi-lane portions of roads, which are common on highways, on main arteries of urban areas, and on approaches to traffic interchanges.

7.2.4 Connections between patterns and their surroundings

In Section 7.2.2, we described basic patterns consisting of a zero-edge subgraph G^0 and left and right subgraphs, G^L and G^R. We assumed that the patterns were connected with the surrounding graph at the ends of G^L and G^R. Specifically, the vertices of G^L and G^R that are not in G^0 may delimit edges that are not in the subgraphs.

Here, we explore the possibility of allowing patterns with additional connections to their surroundings. In particular, we consider two types of connections of a pattern to the surrounding graph. First, the zero-edge graph G^0 may be connected to other subgraphs by zero-edges that do not belong to G^0. Second, edges of the left and the right subgraphs G^L and G^R may be laterally connected to edges that are not in G^L and G^R.

In case of the first type of connection, refinement is not possible. Refining such a subgraph would result in "dangling" zero-edges, i.e., zero-edges exist that are connected to vertices in G^0 that would be eliminated by the refinement.

Also, subgraphs with the second type of connection are not refinable. Refining such a subgraph would result in "partial" lateral connectivity, i.e., an edge exists outside the subgraph to be refined that would be laterally connected to only *part* of an edge that is created by the refinement. Such partial lateral connectivity is not possible in the graph model.

Example 17 Consider the two patterns in the graph in Fig. 15. First, the subgraph G^0 of the simple pattern is connected to the surrounding graph by a zero-edge that does not belong to G^0 (this zero-edge connects to the start of an exit lane). The pattern cannot be refined to a single edge as the connecting zero-edge would be left with a dangling start.

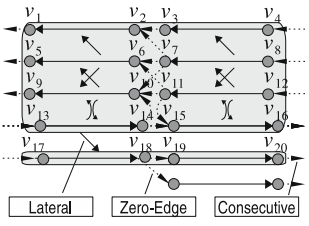


Fig. 15 Pattern connection to the graph

Further, the simple pattern has a lateral connection to the complex one. This also prevents refinement of both patterns. It is not possible to capture in the graph model that parts of edges are laterally connected. Specifically, it is not possible for *parts* of the potential edges $v_{13}v_{16}$ and $v_{17}v_{20}$ to be connected laterally.

Last, both patterns are consecutively connected to edges at their vertices that are not in their G^0 subgraphs. These connections do not affect refinement as the vertices are not modified by any presented refinements.

In summary, we observe that subgraphs with connections other than at their "ends" are not refinable. This explains the choice of basic patterns in Section 7.2.2.

8 Summary and future work

This paper proposes a modeling framework that is intended to serve as a foundation for the provisioning of high-quality, location-based services for road-network constrained mobile users. The framework includes a two-dimensional representation of a transportation network, a graph representation of a transportation network, and a mapping of instances of the former to instances of the latter. The framework also enables the capture of data and query objects.

While the graph representation is compact and is well suited as a basis for the processing of many types of queries, the two-dimensional representation captures the geographical locations of networks and objects, which are necessary for tasks such as the positioning of objects with Euclidean coordinates within a transportation network, known as map matching.

The paper makes three main contributions. The first is the two transportation network representations that are part of the modeling framework. The first representation enables the accurate capture of key aspects of a transportation network. Specifically, the representation enables the capture of lanes and the associated lane-change regulations and u-turn regulations along roads; and it enables the capture of turn restrictions, such as "no left turn" and "no u-turn," at intersections. The representation also captures static objects, e.g., so-called points of interest, with multiple accessibility points and movement-capable objects, e.g., service users, at lane resolution. In addition, the model supports the capture of travel distances and travel times. This detailed level of modeling is increasingly relevant because advances in positioning technologies are slated to enable the positioning of vehicles within lanes. The second representation is a simple directed graph data structure that embodies two non-traditional aspects: it is a multi-graph, and it captures so-called lateral connectivities between edges by means of lateral connectivity functions. These capture the possibilities of making u-turns and lane changes.

The second is the mapping from the rich representation to the graph representation of a transportation network. By devising this mapping, the paper leverages the large body of existing query processing techniques for graphs. This mapping demonstrates constructively how it is possible to model complex road networks by means of (slightly extended) graphs.

The third contribution is a collection of transformations that enable the compacting of graphs. The transformed graphs retain the properties of the source graph that are relevant to the intended applications of the graphs, while reducing the size of the graph.

This work may be extended in several interesting directions. First, the richness of the model may be increased by taking into account additional aspects of road networks, e.g., time-varying properties of the domain, and objects and properties with extent. It may also be of interest to introduce movement functions for the two-dimensional representation of query points. Second, as the model is intended to serve as a basis for query processing in the context of location-based services, it is highly relevant to explore the processing of various proximity queries—conventional as well as trajectory-based—for network-constrained, static and moving objects based on the model. We believe that this may lead to both refinement of the model and new insight into query processing.

Acknowledgements We would like to thank Irina Aleksandrova and Augustas Kligys for their collaborations during the early stages of this work. This work was supported in part by a grant from the Electronics and Telecommunications Research Institute, South Korea.

References

1. V.T. de Almeida and R.H. Güting. "Using Dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases," in *Proc. ACM Symp. on Appl. Comp. (SAC)*, pp. 58–62, Dijon, France, April 2006.
2. Assisted GPS. *http://en.wikipedia.org/wiki/Assisted_GPS*, current as of November, 2006.
3. R. Benetis, C.S. Jensen, G. Karčiauskas, and S. Šaltenis. "Nearest neighbor and reverse nearest neighbor queries for moving objects," in *Proc. Int. Database Eng. Applic. Symp. (IDEAS)*, pp. 44–53, Edmonton, Canada, July 2002.
4. T. Caldwell. "On finding minimum routes in a network with turn penalties," *Communications of the ACM*, Vol. 4(2):107–108, 1961.
5. A. Civilis, C.S. Jensen, J. Nenortaite, and S. Pakalnis. "Efficient tracking of moving objects with precision guarantees," in *Proc. Int. Conf. on Mob. and Ubiq. Syst.*, pp. 164–173, 2004.
6. A. Civilis, C.S. Jensen, and S. Pakalnis. "Techniques for efficient road-network-based tracking of moving objects," in *IEEE Trans. on Knowl. Data Eng.*, Vol. 17(5), pp. 698–712, 2005.
7. E.W. Dijkstra. "A note on two problems in connection with graphs," *Numerische Mathematik*, Vol. 1:269–71, 1959.
8. K. Dueker and J.A. Butler. "GIS-T enterprise data model with suggested implementation choices," *Journal of the Urban and Regional Information Systems Association*, Vol. 10(1): 12–36, 1998.
9. European Geostationary Navigation Overlay Service (EGNOS). "The European space agency," in *http://www.esa.int/esaNA/egnos.html*, current as of November, 2006.
10. P. Fohl, K.M. Curtin, M.F. Goodchild, and R.L. Church, "A non-planar, lanebased navigable data model for ITS," in *Proc. International Symposium on Spatial Data Handling*, pp. 17–29, 1996.
11. R.H. Güting and M. Schneider. Moving Objects Databases. Morgan Kaufmann, 2005.
12. Galileo. "The European Space Agency," in *http://www.esa.int/esaNA/galileo.html*, current as of November 2006.
13. J. Gottsegen, M. Goodchild, and R. Church. "A conceptual navigable database model for intelligent vehicle highway systems," in *Proc. GIS/LIS*, pp. 371–380, 1994.
14. GIS for Transportation (GIS-T) Symposium. *http://www.gis-t.org/*.
15. R.H. Güting, V.T. de Almeida, and Z. Ding. "Modeling and querying moving objects in networks," *VLDB Journal*, Vol. 15(2):165–190, 2006.
16. R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. "A foundation for representing and querying moving objects," *ACM Transactions on Database Systems*, Vol. 25(1):1–42, 2000.
17. C.S. Jensen. "Database Aspects of location-based services," in J. Schiller and A. Voisard (Eds.), *Location-Based Services*, 115–148, Morgan Kaufmann, 2004.

18. G. Kollios, D. Gunopulos, and V.J. Tsotras. "Nearest neighbor queries in a mobile environment," in *Proc. Int. Workshop on Spatio-Temp. Database Management, (STDBM)*, pp. 119–134, Edinburgh, Scotland, September 1999.
19. M. Kolahdouzan and C. Shahabi. "Voronoi-based k nearest neighbor search for spatial network databases," in *Proc. 30th Int. Conf. on Very Large Data Bases (VLDB)*, pp. 840–851, Toronto, Canada, August 2004.
20. K. Mouratidis, M.L. Yiu, D. Papadias, and N. Mamoulis. "Continuous nearest neighbor monitoring in road networks," in *Proc. 32nd Int. Conf. on Very Large Data Bases*, pp. 43–54, Seoul, Korea, September 2006.
21. NCHRP. A Generic Data Model for Linear Referencing Systems, Transportation Research Board: Washington, DC, 1997.
22. Oracle database 10g: oracle spatial network data model, An Oracle Technical White Paper, 2005.
23. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. "Query Processing in spatial network databases," in *Proc. VLDB*, pp. 802–813, 2003.
24. S.F. Rounds, Y. Bock, L. Bock, and J. Fayman. "Epoch-by-EpochTM Real-Time GPS Positioning with the L-3 Communications / Interstate Electronics Corporation GPS Receiver," Presented at the Joint Navig. Conf., 2004.
25. J. Sankaranarayanan, H. Alborzi, and H. Samet. "Efficient query processing on spatial networks," in *Proc. 13th ACM Int. Workshop on Geogr. Inf. Syst. (ACM GIS)*, pp. 200–209, November 2005.
26. C. Shahabi, M.R. Kolahdouzan, and M. Sharifzadeh. "A road network embedding technique for k-nearest neighbor search in moving object databases," in *Proc. 10th ACM Int. Sym. on Adv. in Geogr. Inf. Syst.*, pp. 94–100, McLean, VA, USA, November 2002.
27. A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. "Modeling and querying moving objects," in *Proc. 13th Int. Conf. on Data Eng. (ICDE)*, pp. 422–432, Birmingham, UK, April 1997.
28. S. Šaltenis, C.S. Jensen, S.T. Leutenegger, and M.A. Lopez. "Indexing the positions of continuously moving objects," in *Proc. Int. Conf. on Management of Data (ACM SIGMOD)*, pp. 331–342, Dallas, Texas, USA, May 2000.
29. U.S. Department of Transportation. "Federal-aid highway length—2001 miles by lane width," in *http://www.fhwa.dot.gov/ohim/hs01/hm33.htm*, current as of November 2006.
30. Wide Area Augmentation System (WAAS). "Federal Aviation Administration," in *http://gps.faa.gov/Programs/WAAS/waas.htm*, current as of November 2006.
31. O. Wolfson, L. Jiang, A.P. Sistla, S. Chamberlain, N. Rishe, and M. Deng. "Databases for tracking mobile units in real time," in *Proc., 7th Intt. Conf. Database Theory (ICDT)*, pp. 169–186, Jerusalem, Israel, January 1998.
32. M. Zeiler. Modeling Our World, ESRI Press, 1999.
33. B. Zheng, W.C. Lee, and D.L. Lee. "Search k nearest neighbors on air," in *Proc. 4th Int. Conf. on Mobile Data Management (MDM)*, pp. 181–195, Melbourne, Australia, January 2003.

Laurynas Speičys received the BS degree in Computer Science from Vilnius University, Vilnius, Lithuania in 2000 and the MS degree in Knowledge and Data Engineering from Aalborg University, Aalborg, Denmark in 2002. He is currently working towards his Ph.D. degree in Computer Science

at Aalborg University. Laurynas is a temporary lecturer at the Department of Computer Science of Aalborg University. His research interests are spatial and spatio-temporal databases and conceptual modeling.

Christian S. Jensen received the PhD and DrTechn degrees from Aalborg University, Denmark, in 1991 and 2000, respectively. He is a Professor of Computer Science at Aalborg University and an Adjunct Professor at Agder University College, Norway.

His research concerns data management technology and spans issues of semantics, modeling, and performance. With his colleagues, he receives substantial national and international funding for research, and he has authored or coauthored more than 150 scientific papers.

He is a member of the Board of Trustees of the VLDB Endowment, the EDBT Endowment, and the Danish Academy of Technical Sciences. He received Ib Henriksens Research Award 2001 for his research in mainly temporal data management and Telenor's Nordic Research Award 2002 for his research in mobile services.

His editorial service to journals includes ACM TODS, IEEE TKDE, and the IEEE Data Engineering Bulletin. He was the general chair of the 1995 International Workshop on Temporal Databases and a vice program committee chair for the 1998 IEEE ICDE Conference. He was co-program committee chair for the Workshop on Spatio-Temporal Database Management, held with VLDB'99, and for the 2001 International Symposium on Spatial and Temporal Databases. He was the program committee chair for the 2002 EDBT Conference and the technical program chair for the 2005 VLDB conference. In 2006, he co-chaired the program committee of the 5th International ACM Workshop on Data Engineering for Wireless and Mobile Access. In 2007 he is co-program committee chair for the MDM conference, and in 2008, he is a vice program committee chair for IEEE ICDE.

He serves on the boards of directors and advisors for a small number of technology companies. He serves regularly as a consultant and delivers lectures to industrial and academic audiences.