Propagating and measuring anchor uncertainty in space-time prisms on road networks

Bart Kuijpers[1], Harvey J. Miller[2], Tijs Neutens[3], and Walied Othman[1]

[1] Theoretical Computer Science Group
Hasselt University & Transnational University of Limburg, Belgium
[2] Department of Geography, University of Utah, USA
[3] Department of Geography, Ghent University, Belgium

Abstract. Space-time prisms capture all possible spatio-temporal locations of a moving object between sample points given speed limit constraints on its movement. These sample points are usually considered to be perfect measurements. In this paper we restrict ourselves to a road network and extend the notion of sample points to sample regions, which are bounded, sometimes disconnected, subsets of space-time wherein each point is a possible location, with its respective probability, where a moving object could have originated from or arrived in. This model allows us to model measurement errors, multiple possible simultaneous locations and even flexibility of a moving object.
We develop an algorithm that computes the envelope of all space-time prisms that have an anchor in these sample regions and we developed an algorithm that computes for any spatio-temporal point the probability with which a space-time prism, with anchors in these sample regions, contains that point. We implemented these algorithms in Mathematica to visualise all these newly-introduced concepts.

1 Introduction

One of the dominant ideas underlying the activity-based modelling approach to travel forecasting is that while interacting and performing activities, individuals are faced with the inseparability and scarce nature of space and time. Individual movement implies a trade-off between these resources and is conditioned by the various constraints and opportunities offered by the urban, physical and institutional context in which individuals are embedded [22, 13]. These ideas were already recognised as early as the late 1950s in the seminal work by Hägerstrand [4, 3] and became known as *time geography*. A key concept of time geography is the *space-time prism* which captures potential human movement between two discrete space-time points, also often referred to as anchor points. Classical space time prisms assume a uniform travel velocity in an isotropic and homogenous space. Therefore, they tend to overestimate individual travel possibilities, given that individuals are usually confined to transportation networks with link-specific travel velocities. Hence, they have merely conceptual value but are inadequate for analytically solving problems concerning the extent of individual access. A number of scholars have addressed this issue and

Dagstuhl Seminar Proceedings 08471
Geographic Privacy-Aware Knowledge Discovery and Delivery
http://drops.dagstuhl.de/opus/volltexte/2009/2007

proposed geo-computational algorithms to implement space-time prisms within networks relying on shortest path calculations [11, 28, 10, 9], isochrones [17, 15], and velocity fields [14].

These network-based implementations of space-time prisms assume that the spatio-temporal coordinates of the anchor points are exactly known and strictly fixed in space as well as in time. In reality, however, this is typically not the case as uncertainty may arise from different sources related to the purpose-specific ways in which anchor points of space-time prisms are commonly defined or measured. Firstly, in research on individual space-time accessibility, it has become common practice to define anchor points as the start and end points of fixed activities and to derive these retrospectively from activity and travel diary data [24, 8]. As noted by Rietveld [20] and Witlox [26], the accuracy of reported departure and arrival points is affected by rounding and geocoding imprecision which might bias the assessment of individual accessibility. Secondly, individual movement can be recorded by means of discrete time stamped anchor points obtained with high resolution using location aware technologies (LATs) [6] such as the global positioning system (GPS) and radio-location methods [12]. The use of such devices implies both measurement and sampling errors. While sampling errors relate to the observation frequency with which the anchor points are recorded, measurement errors result from the inaccuracy inherent to the tracking technique used. In the case of GPS, measurement errors are often modelled as a bivariate normal distribution in the (x, y)-plane [19]. Thirdly and perhaps most problematic for representing an individual's travel possibilities is the uncertainty on the part of the individuals themselves. When making scheduling decisions, individuals have to reckon with the uncertain performance of the transportation system and the possibly uncertain duration and location of their future space-time requirements [7]. Finally, as argued by Schwanen [21], considering anchor points as temporally fixed is questionable as arriving on time at a meeting appointment is often conceived of as a range of acceptable arrival times rather than a single clock time.

Despite the variety of sources from which uncertainty in anchor points may arise, until now the main focus has been directed towards the management of uncertainty in moving object databases (MOD) [27, 23, 18] and the effects of uncertain travel times due to systematically recurring congestion or unreliable transportation systems [5, 2]. The equally important question as to what extent uncertain anchor points might influence the possibilities for travel and activity participation, however, has received only scant attention in literature on individual space-time accessibility analysis. This issue pertains to the degree of flexibility of the temporal boundaries of activities. Notable exceptions include Hendricks [7] and Neutens [16], both relying on classical time geography and consequently assuming an isotropic and homogenous travel environment.

This paper presents a formal framework to model space-time prisms on road networks with uncertainty about the anchor points. Although uncertainty falls apart in different subcategories (e.g., inaccuracy, incompleteness, inconsistency, imprecision, vagueness), we will concentrate only those forms of spatio-temporal

uncertainty that can be modelled by two independent probability functions, for space and time respectively. The approach presented allows gaining sound insight in the travel and meeting possibilities of individuals facing uncertain space-time commitments. Anchor points of space-time prisms will be represented by a finite set of possible outcomes to which a degree of uncertainty is assigned. Road networks will be represented by a graph embedding in $\mathbf{R}^2$ that is comprised of edges and vertices labelled with law-imposed speed limits. This paper has a dual aim. Firstly, we will provide and implement an algorithm to compute and visualise space-time prisms on road networks when uncertainty distributions are assigned to the anchor points. Our algorithm computes and visualises the fraction of space-time prisms from these uncertain anchor regions that cover that point. The spatial projection of the network-based space-time prisms under uncertainty is derived as well. Secondly, we will illustrate the usefulness of the implemented algorithm in assessing the implications of uncertain anchor points on the opportunities for individual and joint activity participation.

The remainder of the paper is organised as follows. In Section 2, we start with defining space-time prisms, trajectories, road networks, and trajectories and space-time prisms on road networks. Section 3 then introduces the notion of sample regions and corresponding uncertain space-time prisms. In this section we also present an algorithm to compute and visualise this envelope of an uncertain space-time prism. Section 4 introduces the algorithm to compute the fraction of the uncertain prism that covers a given spatio-temporal point and the probability of that fraction. Moreover, we implemented this in Mathematica to visualise uncertain space-time prisms and the probability in each spatio-temporal point of this prism. In Section 5 we illustrate some immediate applications of our results. Finally, we conclude with the major findings and outline the avenues for future research.

2 Preliminaries

In this section, we outline some basic definitions of the tools we will be using. As we do not introduce new results for the general setting, we restrict ourselves to a road network.

Definition 1. *Let* $\mathbf{R}$ *denote the set of real numbers. A* road network RN *is a graph embedding in* $\mathbf{R}^2$ *of a labelled graph given by a finite set of vertices* $\mathsf{V} = \{(x_i, y_i) \in \mathbf{R}^2 \mid i = 1, \ldots, N\}$ *and a set of edges* $\mathsf{E} \subseteq \mathsf{V} \times \mathsf{V}$ *that are labelled by a* speed limit *and an associated* time span. *This graph embedding satisfies the following conditions. Edges are embedded as straight line segments between vertices.*[4] *If an edge between* (x_i, y_i) *and* (x_j, y_j) *is labeled by the speed limit* $v_{ij} > 0$*, then its time span* w_{ij} *is* $\frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{v_{ij}}$*, i.e., it is the time needed to get from one side of an edge to another when travelling at the speed limit.* □

[4] These edge embeddings may intersect in non-vertex points. So, we can model bridges and tunnels in our model.

So, we have $\mathsf{RN} = \{(x, y) = (1 - \lambda)(x_i, y_i) + \lambda(x_j, y_j) \mid \lambda \in [0, 1]$ and $((x_i, y_i), (x_j, y_j)) \in \mathsf{E}\} \cup \mathsf{V}$.

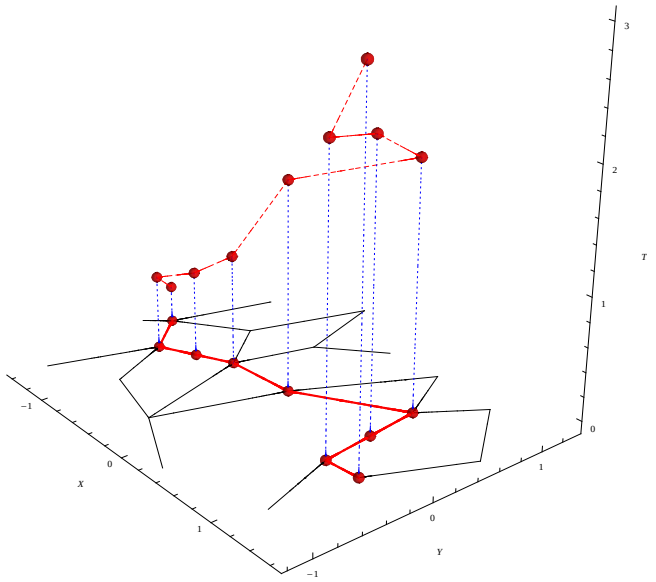

Fig. 1. A trajectory in space-time and its projection on the road network.

A trajectory can be anything continuous, be it a polyline or something more differentiable. We assume trajectories are recorded as a discrete list of time-stamped locations, called sample points, and that nothing is known about an object's position between those sample points other than an upper bound on its speed. Moreover, we assume all our trajectories to be part of the road network.

Definition 2. *First we define trajectories and then their restriction to a road network.*

- Let $I \subseteq \mathbf{R}$ be an interval. A *trajectory* T is the graph of a mapping $\alpha : I \to \mathbf{R}^2 : t \mapsto \alpha(t) = (\alpha_x(t), \alpha_y(t))$, i.e., $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$. We call I the *time domain* of T.
- A *trajectory sample* is a finite set $S = \{(t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\}$ of time-space points. The order on time, $t_0 < t_1 < \cdots < t_N$, induces a natural order on the sample.
- If T is a trajectory given by the functions α_x and α_y, then it must satisfy $(\alpha_x(t), \alpha_y(t)) \in \mathsf{RN}$ for all t in the time domain of T and for a trajectory sample $S = \{(t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\}$ we must have $(x_i, y_i) \in$

RN for all $i = 0, \ldots, N$. A trajectory (sample) on a road network RN is a trajectory (sample) whose spatial projection is in RN, as illustrated in Figure 1.

□

Figure 1 shows a road network and a trajectory, as well as a sample, on top of it.

We now define space-time prisms. In the traditional sense, a space-time prism is the union of all trajectories from one point to another that are constraint by a speed limit. A space-time prism for an object moving in the real plane, unconstrained in its movement, is defined as follows [4, 12, 1].

Definition 3. *Let $p = (x_p, y_p)$, $q = (x_q, y_q) \in \mathbf{R}^2$. The set of points (t, x, y) in the space-time prism with origin (t_p, p), destination (t_q, q) and maximal speed $v_{\max} > 0$ satisfy the constraints*

$$\begin{cases} (x - x_p)^2 + (y - y_p)^2 \leq (t - t_p)^2 v_{\max}^2 \\ (x - x_q)^2 + (y - y_q)^2 \leq (t_q - t)^2 v_{\max}^2 \\ t_p \leq t \leq t_q \end{cases}$$

This space-time prism is visualised in Figure 2 and denoted by $\mathcal{P}^{\mathbf{R}^2}(t_p, x_p, y_p, t_q, x_q, y_q, v_{\max})$. □

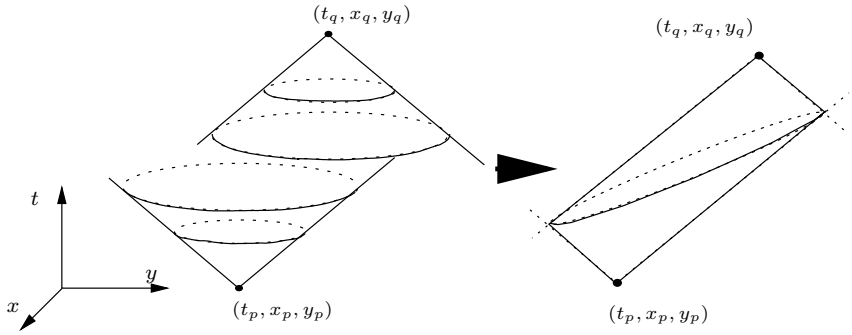

Fig. 2. An example of a space-time prism $\mathcal{P}^{\mathbf{R}^2}(t_p, x_p, y_p, t_q, x_q, y_q, v_{\max})$.

As a side note, a chain of space-time prisms between a list of sample points is called a *lifeline necklace* [1], see Figure 3.

We adapted these space-time prisms to road networks with either uniform speed limits over the entire network or speed limits that may vary per edge, and for bi- or unidirectional edges. A space-time prism for an object moving on a road network is not the intersection of a cylinder on top of a road network and the unconstrained space-time prism.

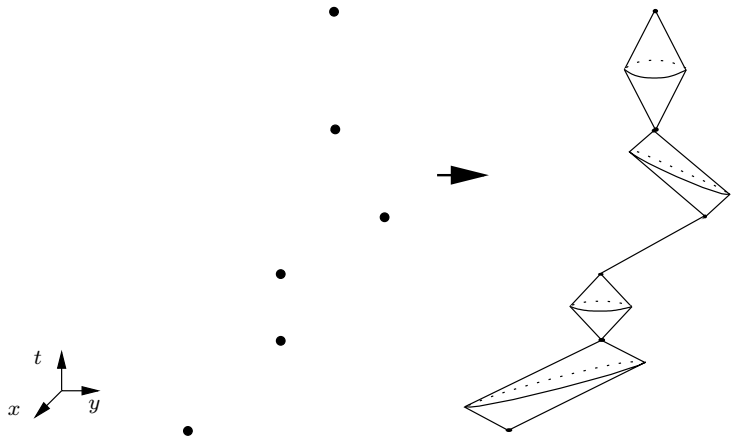

Fig. 3. A trajectory sample and a lifeline necklace.

To define space-time prisms on a road network, we need to define an appropriate distance function on the network. This distance measure that we use is derived from the *shortest path*-distance used in graph theory [25].

Definition 4. *Let* RN *be a road network,* $p, q \in \mathsf{RN}$ *and let* V_{pq} *be the set of vertices* $\mathsf{V} \cup \{p, q\}$ *and let* E_{pq} *equal the set of edges obtained from* E *by adding* p *and* q *to the vertex set. The* road network time *between* p *and* q*, denoted by* $\mathsf{d}_{\mathsf{RN}}(p, q)$*, is the shortest-path distance*[5] *between* p *and* q *in the graph* $(\mathsf{V}_{pq}, \mathsf{E}_{pq})$*, with respect to the time-span labelling of the edges.*

A path from p *to* q *whose length, with respect to the time-span labelling of the edges, equals* $\mathsf{d}_{\mathsf{RN}}(p, q)$ *is called a* fastest path from p to q. □

Note that the *road network time* between p and q in the above definition has minimal total weight and equals the shortest time span in which you can reach q from p. The metric that we describe takes two points from a road network and returns the shortest time needed to get from one to the other when travelling at the allowed maximal speed at each segment.

A space-time prism on a road network is the geometric location in $\mathbf{R} \times \mathsf{RN} \subset \mathbf{R} \times \mathbf{R}^2$ of all points a moving object could have visited when travelling, restricted to RN, from an origin p to a destination q with in a time-frame ranging from t_p to t_q, respecting the speed limits on the edges of RN. We define this more formally. Given a road network RN, points p, q and u on RN, and time moments t_p and t_q for p and q, we write t_u^- to abbreviate $t_p + \mathsf{d}_{\mathsf{RN}}(p, u)$ and t_u^+ to abbreviate $t_q - \mathsf{d}_{\mathsf{RN}}(u, q)$.

Definition 5. *Let* RN *be a road network, let* $p, q \in \mathsf{RN}$*. The* space-time prism on the road network *between* (t_p, p) *and* (t_q, q)*, with respect to the speed limits of*

[5] We mean the single-pair shortest-path distance that is commonly used in graph theory and that can be computed efficiently by the well known Dijkstra's algorithm [25].

RN *is denoted by* $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ *and is defined as the set of* (t, u) *tuples, where* $u \in \mathbf{R}^2$ *for which*

$$\begin{cases} u \in \mathsf{RN}, \\ \mathsf{d}_{\mathsf{RN}}(p, u) + \mathsf{d}_{\mathsf{RN}}(u, q) \leq (t_q - t_p), \\ t_u^- \leq t \leq t_u^+. \end{cases}$$

The space-time prism on the road network *between* (t_p, p) *and* (t_q, q)*, with respect to a general maximal speed* $v_{\max}$ *is the space-time prism on* RN *after relabelling all edges of* RN *with speed limit* $v_{\max}$. □

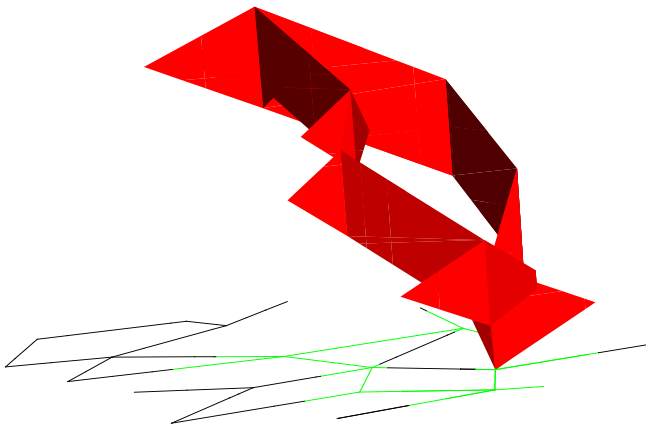

Fig. 4. A road network space-time prism and its projection on the road network.

Figure 4 illustrates what these space-time prisms on road networks look like. Figure 4 was generated using our own Mathematica implementation, which is available at [**?**].

In the remainder of the paper we will restrict ourselves to the road network setting.

3 From uncertainty on sample points to sample regions and uncertain space-time prisms

In the preliminaries, the sample points were considered perfect data and represented by space-time points. In real life however, a lot of sources introduce errors on sample points. Measurement errors, for one, are introduced when measuring locations using GPS, for example. When a human is asked to keep track of its locations and time spent there, errors get easily introduced, e.g., "I left work between 5 and 5:30pm". Therefore, we extend the concept of certain sample points to sample regions.

3.1 uncertain sample points

In this section, we expand our model to the use of sample regions to supersede the notion of sample points. For simplicity, we will start by defining a sample region on a line segment and later expand this definition to road networks. Another simplification in our model is that time and space are considered independent from each other. This ensures that our sample regions are box-shaped in space-time and that our model behaves in a polygonal fashion. We coin this rectangle a *sample region*.

Moreover, in this sample region some subsets can be more likely than others. This can be described using probability functions. We refer to Figure 5 for a conceptual representation of this model.

Definition 6. *A* sample region on a segment *is a bounded subset of space-time of all possible locations of a sample point. Let* $p = (x_p, y_p), q = (x_q, y_q) \in \mathbf{R}^2$ *be the spatial borders and* $t_{pq}^-, t_{pq}^+ \in \mathbf{R}$*, where* $t_{pq}^- \leq t_{pq}^+$*, the temporal borders of the region, meaning the region is bounded by the polygon* $\langle (t_{pq}^-, x_p, y_p), (t_{pq}^+, x_p, y_p), (t_{pq}^+, x_q, y_q), (t_{pq}^-, x_q, y_q) \rangle$*. We encode this region by the* 6*-tuple* $S_{pq} = (p, q, t_{pq}^-, t_{pq}^+, \mu_{pq}, \chi_{pq})$*, where* $\chi_{pq} : \mathbf{R} \to \mathbf{R}^+$ *and* $\mu_{pq} : [t_{pq}^-, t_{pq}^+] \to \mathbf{R}^+$ *are independent probability functions and* $\chi_{pq} : \lambda \mapsto \chi_{pq}(\lambda)$ *where* χ_{pq} *is a probability function on the line* $(x, y) = (1 - \lambda)p + \lambda q$*.* □

Note that in the definition above we do not demand that χ_{pq} is restricted to $[0, 1]$. The reason for this is that on a road network RN we will stitch a finite number of these previously defined sample regions together and we want these probability functions to integrate to one on all these regions combined.

Definition 7. *A* sample region S on a road network RN *is a bounded subset of space-time of all possible locations of a sample point on* RN*. In particular, it is a finite set of* sample regions on segments *as defined in Definition 6,* $S = \cup_i S_i$ *where* $i = 1, \ldots, n$ *and each* S_i *is a sample region on a segment. Moreover, all the* S_i *are disjoint and integrating all the spatial probability functions over all the spatial component of these sample regions adds up to one.* □

This is illustrated in Figure 5. An *uncertain trajectory sample* is then a finite list of sample regions and constitutes a new definition of trajectory samples.

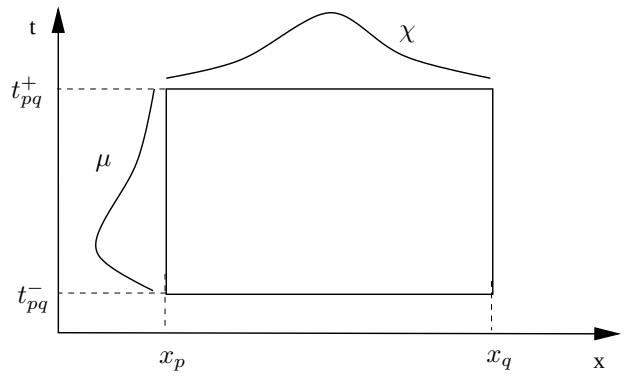

Fig. 5. An example of a sample region.

Remark 1. – Note that we are not demanding that the regions stitch nicely together in time. This allows us to model locations that are accessible at discrete intervals in time. For example, shopping malls that are not open 24 hours a day.
– Secondly, we do not demand that they are continuous in space either. This, in turn, allows us to model several probable departure and arrival locations, and, as we will soon elaborate on, calculate a relative likelihood for each region.

3.2 uncertain space-time prisms

The next step is to adapt the space-time prism model to these sample regions. This can be done in a straightforward manner as described in the next definition.

Definition 8. *Let* RN *be a road network and S_b and S_e sample regions on* RN*. An* uncertain space-time prism *between the sample regions S_b and S_e is the union of all space-time prisms with starting point in S_b and ending point in S_e. An additional constraint is that there needs to exist a space-time prism from every point in S_b that has an endpoint in S_e and vice versa.* □

When every point in a sample region is equally probable then the distribution function for time and space is a uniform distribution. To model those points that are more probable around the center of a sample region than at the edges, a normal distribution can be used.

3.3 Computing the envelope of the uncertain prism

In this section we introduce an algorithm that computes the uncertain space-time prism, which envelopes the union of all space-time prisms that connect a point from the starting sample regions to a point in the ending sample regions. This algorithm is a slight adaptation of earlier work [9] where also the proofs of

the correctness of this algorithm can be found. We consider each edge separately and it suffices that we compute, for both vertices of that edge, the earliest arrival times and latest departure times.

The following observations will simplify the computations significantly. For each edge, where we are constructing the uncertain prism, we cycle through all the edges of the starting sample region. For each such edge we note that the fastest path has to pass over one of the nodes of that edge. Moreover, a path from the interior of such an edge is longer than a path that departs from at least one of the nodes. So the earliest time you can reach a node is the minimum of the road network distances to that node from all nodes of the departure sample region at the earliest time of that region. Likewise, the latest departure times at each node will equal the maximum of all road network distances to all nodes of the arrival sample region at the latest time of that region.

The algorithm looks much like the one we presented in [9] . The major difference is the pre-computation part of the algorithm. In the first step we use an adapted breadth-first search to pre-select the vertices that can be part of the space-time prism and ignore the rest. In the second step we compute the earliest arrival and latest departure times for each vertex. This computation is not sufficient for the edges that support sample regions, since these computations that do not take the points in the interior into account.

PRECOMP: input= $(\mathsf{V}, \mathsf{E}, S_b, S_e)$;
output= $(\mathsf{RN}', \{(t_u^-, t_u^+) \mid u \in \mathsf{RN}'\}, \mathsf{V}_\mathcal{P}, \mathsf{E}_\mathcal{P})$

Step 1. In this step we add vertices to the network and select those nodes and vertices that can actually support the prism.

- For all $S_{b,i}, S_{e,j}$, say such a region is spatially bounded by $p, q \in \mathbf{R}^2$. If p is a vertex then do nothing, else let $r, s \in \mathbf{R}^2$ be the vertices that bound the edge $[r, s]$ that contains p. Remove that edge from the network, add the vertex p to the network and add the edges $[r, p]$ and $[p, s]$ to the network. Repeat the same procedure for q.
- The second part of this step consists of an adapted breadth-first search algorithm where we keep looking for and storing vertices and edges until their road network distance to any $S_{b,i}$ or $S_{e,j}$ is larger than the maximal difference in time between any pair $(S_{b,i}, S_{e,j})$. Let $t_{\max} = \max_{i,j}\{t_{e,j}^+ - t_{b,i}^- \mid S_{b,i} = (p_{b,i}, q_{b,i}, t_{b,i}^-, t_{b,i}^+, \mu_{b,i}, \chi_{b,i})$ and $S_{e,j} = (p_{e,j}, q_{e,j}, t_{e,j}^-, t_{e,j}^+, \mu_{e,j}, \chi_{e,j})\}$. The following steps need to be repeated for all $\{r \mid \exists i : S_{b,i} = (p, q, t^-, t^+, \mu, \chi)$ or $\exists j : S_{e,j} = (p, q, t^-, t^+, \mu, \chi)$ and $r = p$ or $r = q\}$.
  Initialise a queue with the node r and distance 0. Add r to the road network RN'. Repeat the following steps until the queue is empty.
  1. Remove the top element from the queue, which is a vertex s and a distance t_s.
  2. For all the edges $[p, s]$ connected to s that are not handled yet do the following:
     - Add the vertex p and the edge $[p, s]$ to RN' and mark the edge $[p, s]$ as handled.

- **If** $t_s + \mathsf{d}_{\mathsf{RN}}(s,p) \leq t_{\max}$ **then** add the vertex p and the distance $t_s + \mathsf{d}_{\mathsf{RN}}(s,p)$ to the queue.

Step 2. In this step we compute the earliest arrival time and latest departure time in each vertex, with respect to all the sample regions where we can leave from and all regions where we can arrive in.

Let $V_b = \{(r,t^-) \mid \exists i : S_{b,i} = (p,q,t^-,t^+,\mu,\chi)$ and $r = p$ or $r = q\}$ and $V_e = \{(r,t^+) \mid \exists j : S_{e,j} = (p,q,t^-,t^+,\mu,\chi)$ and $r = p$ or $r = q\}$.

Now we cycle through all the pairs (r,t^-) contained in V_b and apply a single-source shortest path algorithm (e.g., Dijkstra's algorithm) for each r on the graph RN'. For each vertex u in RN' we store its smallest road network time from r, i.e., the arrival time $t_u^- = t^- + \mathsf{d}_{\mathsf{RN}}(r,u)$. For the first node r we initialise the nodes u with that arrival time, for all other nodes r we set the arrival time t_u^- to $t_u^- = \min\{t_u^-, t^- + \mathsf{d}_{\mathsf{RN}}(r,u)\}$.

Again, we cycle through all the pairs (r,t^+) contained in V_e and apply a single-source shortest path algorithm (e.g., Dijkstra's algorithm) for each r on the graph RN'. For each vertex u in RN' we store its largest road network time from r, i.e., the arrival time $t_u^+ = t^+ - \mathsf{d}_{\mathsf{RN}}(r,u)$. For the first node r we initialise the nodes u with that arrival time, for all other nodes r we set the arrival time t_u^+ to $t_u^+ = \max\{t_u^+, t^+ - \mathsf{d}_{\mathsf{RN}}(r,u)\}$.

When we make our last pass for the last element of V_e, we store each vertex u for which $t_u^- \leq t_u^+$ in the set $\mathsf{V}_{\mathcal{P}}$. In the set $\mathsf{E}_{\mathcal{P}}$ we store all edges that connect to at least one vertex in $\mathsf{V}_{\mathcal{P}}$.

The next step is to construct the polygons that constitute the space-time prism. This step is identical to the algorithm 3D-SPACE-TIME PRISM described in [9]. Since we already provided a correctness proof in that same paper we will omit to repeat this here. The only difference is that we need to correct the polygons for the sample regions since our computations neglected the internal points of the sample region. An example of this is illustrated in Figure 6.

We can correct this by adding an extra, easy to compute, triangular polygon for each sample region. For each starting region $S_{b,i} = (p,q,t^-,t^+,\mu,\chi)$ we add the polygon $\langle (p,t^-),(q,t^-),(\frac{p+q}{2},\frac{\mathsf{d}_{\mathsf{RN}}(p,q)}{2})\rangle$ as illustrated in Figure 6 on the right. Likewise, for each ending region $S_{e,j} = (p,q,t^-,t^+,\mu,\chi)$ we add the polygon $\langle (p,t^+),(q,t^+),(\frac{p+q}{2},-\frac{\mathsf{d}_{\mathsf{RN}}(p,q)}{2})\rangle$.

3D-SPACE-TIME PRISM: input= $(\mathsf{RN}', \{(t_u^-, t_u^+\} \mid u \in \mathsf{RN}'\}, \mathsf{V}_{\mathcal{P}}, \mathsf{E}_{\mathcal{P}})$;
output= drawing of $\mathcal{P}^{\mathsf{RN}}(t_p,p,t_q,q,)$.

For each edge (r,s), with $r = (x_r,y_r)$ and $s = (x_s,y_s)$, in E_{B}, we do the following for r and s. There are several possible cases one needs to consider and they are illustrated in Figure 7 and Figure 8.

– These are Cases 1 and 2, illustrated in Figure 7: If $w_{rs} \leq \frac{(t_r^+ - t_r^-)}{2}$, then draw the polygon $\langle (t_0,x_0,y_0),(t_r^+,x_r,y_r),(t_r^-,x_r,y_r),(t_0,x_0,y_0)\rangle$ where $t_0 =$

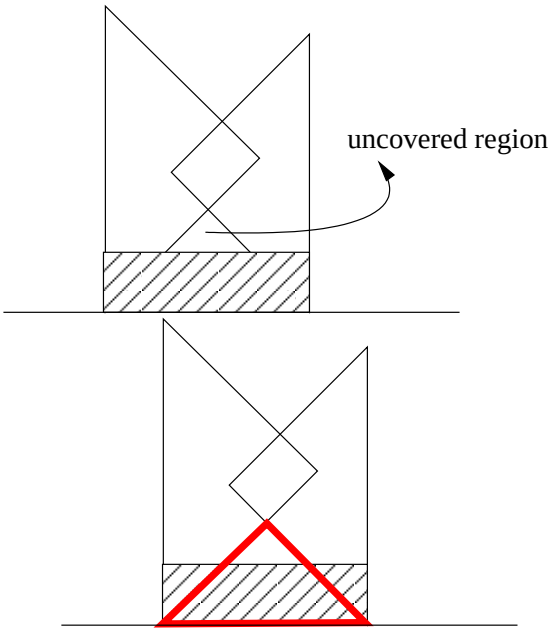

Fig. 6. The omitted part of a sample region.

$\frac{t_r^- + t_r^+}{2}$ and

$$(x_0, y_0) = (x_r, y_r) + \frac{(t_r^+ - t_r^-)}{2} v_{rs} \frac{(x_s - x_r, y_s - y_r)}{d_{rs}}.$$

Otherwise, draw the polygon $\langle (t_1, x_s, y_s), (t_r^+, x_r, y_r), (t_r^-, x_r, y_r), (t_0, x_s, y_s), (t_1, x_s, y_s) \rangle$ where $t_0 = t_r^- + w_{rs}$ and $t_1 = t_r^+ - w_{rs}$.

– This is Case 3, illustrated in Figure 8 in the left most figure: In this case one is able to reach s from r before one has to leave s again. In other words, $t_s^+ \geq t_r^- + w_{rs}$. In which case the polygon $\langle (t_r^-, x_r, y_r), (t_1, x_r, y_r), (t_s^+, x_s, y_s), (t_0, x_s, y_s) \rangle$ is drawn where $t_0 = t_r^- + w_{rs}$ and $t_1 = t_s^+ - w_{rs}$.

  Cases 1, 2 and 3 combined gives us the two right most figures of Figure 8.

– Repeat the previous steps with the indices r and s interchanged.

The result of this pre-computation algorithm, together with the 3D-SPACE-TIME PRISM-algorithm is shown in Figure 9. Note that unlike in Figure 4, where a prism starts and ends in a single point, this prism has a lot of possible locations to start from and to arrive at.

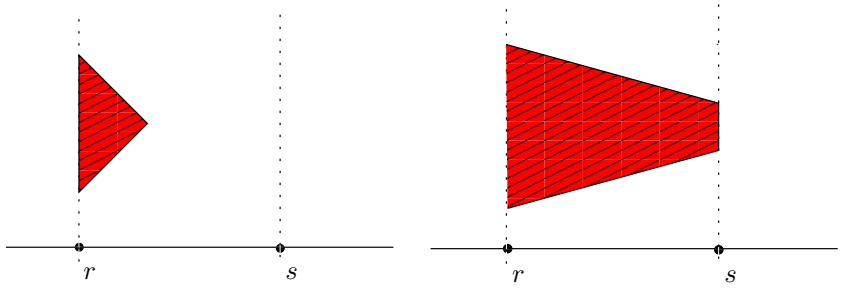

Fig. 7. Cases 1 and 2.

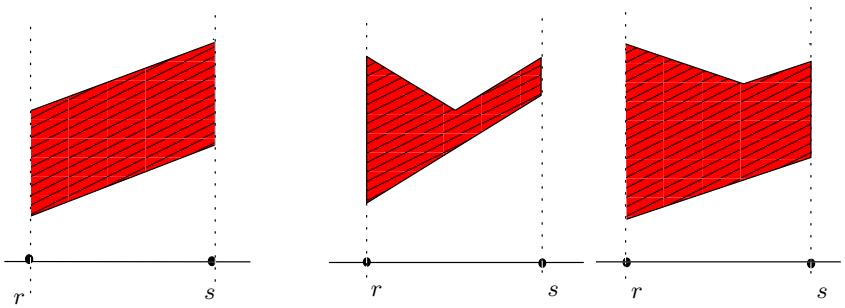

Fig. 8. Cases 3, 4 and 5.

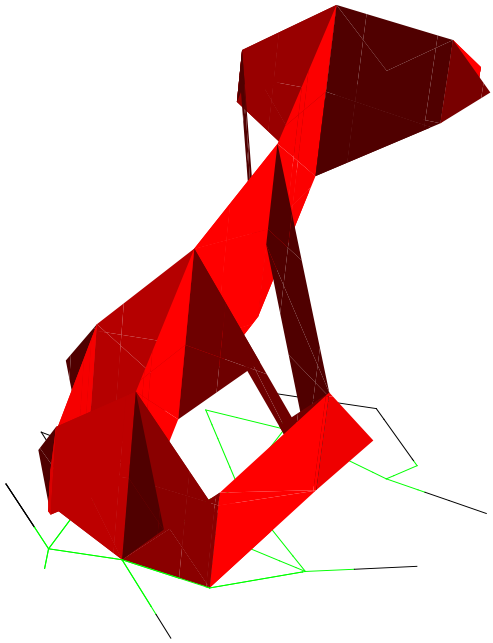

Fig. 9. An envelope of space-time prisms on sample regions.

4 Measuring spatio-temporal uncertainty and flexibility with respect to sample regions

In this section we take things a step further and introduce the main contribution of this paper. When we introduced sample regions, we used distributions functions to model the likelihood of every point in the sample region, but we have yet to exploit those attributes of sample regions.

Let **r** be any spatio-temporal point inside the envelope of the uncertain space-time prism. We know that **r** is covered by at least one space-time prism with a starting point in the starting region and an ending point in the ending region. Suppose, and this is the case for most points in the envelope, that there is an entire subset of the sample regions consisting of points which are anchors for *space-time prisms that contain* **r**. We can then *measure*,i.e., integrate the distribution functions over, these subsets by means of the distribution functions we introduced in Definition 6, and we can choose to

- restrict ourselves to the starting regions,
- restrict ourselves to the ending regions,

– multiply the two numbers above.

In all three cases we get a number between zero and one, this is per construction of the sample regions.

In the first case we obtain the likelihood that the anchor of a space-time prism that contains $\mathbf{r}$ is part of the starting regions. In the second case we obtain the likelihood that the anchor of a space-time prism that contains $\mathbf{r}$ is part of the ending regions. In the third case we obtain the simultaneous likelihood that the anchor of a space-time prism that contains $\mathbf{r}$ is part of the starting and ending regions. The fraction of the sample regions that have a starting and ending point of a space-time prism that contains that particular space-time point $\mathbf{r}$.

Definition 9. *Let* $\mathbf{r}$ *be a spatio-temporal point on a road network* RN *and* S_b, S_e *sample regions on* RN.

- The emanating fraction of $\mathbf{r}$ with respect to S_b *equals the measure, with respect to the distribution functions of* S*, of the subsets of* S *that contain anchors, with a* smaller *time coordinate than* $\mathbf{r}$ *of space-time prisms on* RN *that contain* r*.*
- The absorbing fraction of $\mathbf{r}$ with respect to S_e *equals the measure, with respect to the distribution functions of* S*, of the subsets of* S *that contain anchors, with a* larger *time coordinate than* $\mathbf{r}$ *of space-time prisms on* RN *that contain* r*.*
- The fraction of $\mathbf{r}$ with respect to travel from S_b to S_e *equals the product of (the emanating fraction of* $\mathbf{r}$ *with respect to* S_b*) with (the absorbing fraction of* $\mathbf{r}$ *with respect to* S_e*).*

□

In the following section we show the surprisingly simple, i.e., polygonal, shape of these subsets. Moreover we provide an algorithm to compute these subsets and associated fractions as defined in Definition 9.

4.1 Algorithm(s)

The algorithm is based on the following observations, which do not require proof.

Due to the additive nature of integrals, i.e., an integral over a surface equals the sum of integrals over disjoint subsets that cover the surface, we can treat each of the rectangular regions in space-time separately and add them together once the computation is done. A sample region can thus be seen as the sum of sample regions on straight edges of the road network.

Let $\mathbf{r} = (t_r, x_r, y_r)$ be the space-time point for which we wish to compute the fraction of space-time prisms that contain that point. For each of those separate rectangular sample regions we can distinguish between two cases. Either there exists a point s in the spatial part of the sample region for which there exists more than one fastest path to $\mathbf{r}$, or there does not. If this point exists we simply divide the rectangular region into two new regions along the temporal line through s. This operation ensures we are again in the latter case, where there exists no

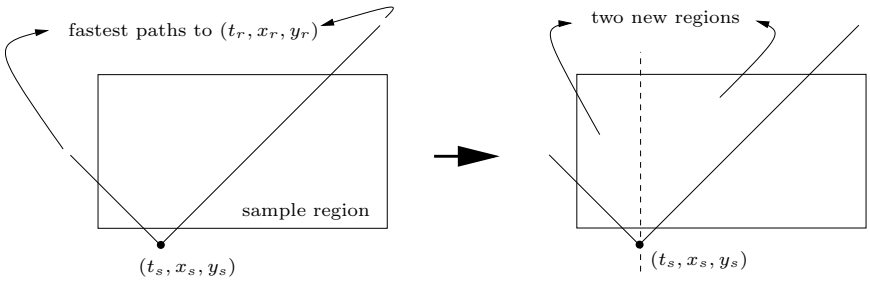

Fig. 10. A divided sample region.

spatial point, in the interior of the region, for which there exists more than one fastest path to **r**. See Figure 10.

The second observation, which we will prove in Theorem 1, states that it suffices to find all space-time paths that originate in the starting sample region and end in r, and all space-time paths that originate in r and arrive in the ending sample region.

Theorem 1. *There exists a space-time prism* $\mathcal{P}^{\mathsf{RN}}(t_p, x_p, y_p, t_q, x_q, y_q)$ *that contains* **r** if and only if *there exists a trajectory from* (t_p, x_p, y_p) *to* (t_q, x_q, y_q) *on* RN *through* **r**.

Proof. The proof is trivial, the equivalence holds by the very definition of a space-time prism.

Though the proof is trivial, it holds the key to our algorithm, which is based on the following two observations. Assume that the interior of the sample region S_b does not contain a point with more than one path with the same road network time to r. We will show in the algorithm below how to split the sample regions to smaller regions that satisfy that condition.

1. First, a fastest path that leads to **r** and contains the edge contained by S_b either intersects S_b, goes over S_b, i.e., all its time coordinates are larger than those of S_b, or under S_b, i.e., all its time coordinates are smaller than those of S_b. If it goes over S_b, i.e., all its time coordinates are greater than those of S_b, then all points of S_b clearly have a path to **r**, since any moving object in a point in S_b can just wait until its time coordinate equals that of the fastest path and leave to **r** following that path. Likewise, if the path goes under S_b, i.e., all its time coordinates are smaller than those of S_b, then no moving object departing from S_b is able to reach **r** in time. If the path intersects S_b, then all space-time points of S_b with a time coordinate smaller than the point on the path with the same spatial coordinates have a path that reaches **r** in time. This is depicted in the shaded region in Figure 11a.
2. Secondly, a fastest path that emanates from **r** and contains the edge contained by S_e either intersects S_e, goes over S_e or under S_e. If it goes under

S_e, i.e., all its time coordinates are smaller than those of S_e, then all points of S_e can be reached from $\mathbf{r}$, since any moving object that departs from $\mathbf{r}$ can reach a point with spatial coordinates in S_e and wait until its time coordinate equals that of a point in S_e. Likewise, if the path goes over S_e, i.e., all its time coordinates are greater than those of S_e, then no moving object departing from $\mathbf{r}$ is able to reach S_e in time. If the path intersects S_e, then all space-time points of S_e with a time coordinate greater than the point on the fastest path with the same spatial coordinates have a path from $\mathbf{r}$ that can be reached in time. This is depicted in the shaded region in Figure 11b.

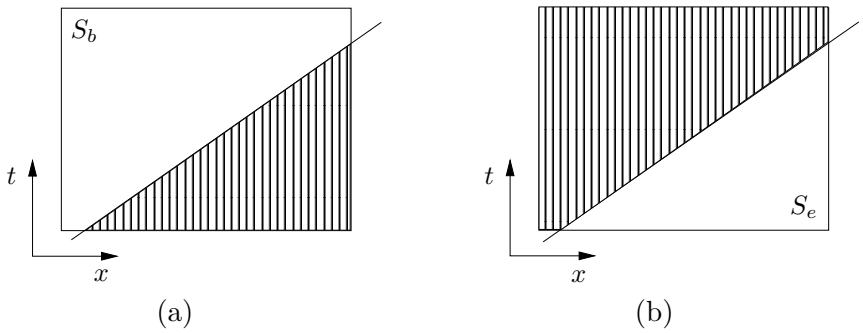

Fig. 11. Subsets of sample regions that are possible to connect to a fixed space-time point.

These observations are necessary and sufficient to compose the algorithm. Once these areas have been determined we can integrate the distribution functions over them and compute a probability for a specific space-time point. The nature of these areas allow easy computation of these integrals.

Let $r = (t_r, x_r, y_r)$ be the space-time point for which we wish to compute the fraction of space-time prisms covering it. Let S_b be the starting sample region and S_e be the ending sample region. Let $S_{b,i}$ be the restriction of S_b to a single edge and i be a natural number to count the number sample regions on $\mathbf{R}$ that S_b contains, the definition of $S_{e,i}$ is analogous. The first step in the algorithm is to compute appropriately sized sample regions. These are regions where each point in the spatial interior has a unique fastest path to the given space-time point $\mathbf{r}$.

The following algorithm takes a spatio-temporal point $r = (t_r, x_r, y_r)$ as input and outputs three *numbers*: the emanating fraction $\mathcal{S}_{b,r}$ of $\mathbf{r}$ with respect to S_b, the absorbing fraction $\mathcal{S}_{e,r}$ of $\mathbf{r}$ with respect to S_e and the fraction $\mathcal{S}_r$ of $\mathbf{r}$ with respect to travel from S_b to S_e.

PointProbability: input= $(\mathsf{V}, \mathsf{E}, t_r, x_r, y_r, S_b, S_e)$; output= $\mathcal{S}_{b,r}, \mathcal{S}_{e,r}, \mathcal{S}_r$

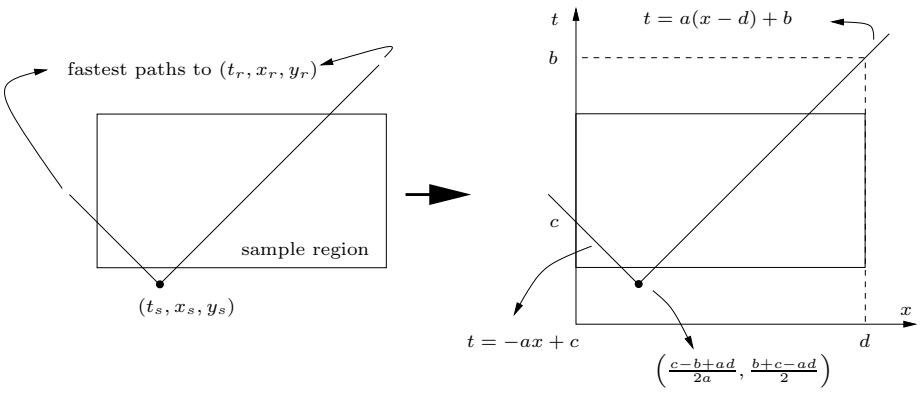

Fig. 12. How to split a sample region.

Initialisation. Set $\mathcal{S}_{b,r}$ and $\mathcal{S}_{e,r}$ equal to 0.

Step 1. Let $S_{b,i} = (p, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq})$ where $p = (x_p, y_p), q = (x_q, y_q)$. The following needs to be repeated for each $S_{b,i}$. Let $d_p = t_r - \mathsf{d}_{\mathsf{RN}}((x_r, y_r), (x_p, y_p))$, $d_q = t_r - \mathsf{d}_{\mathsf{RN}}((x_r, y_r), (x_q, y_q))$, $d_{pq} = \mathsf{d}_{\mathsf{RN}}((x_p, y_p), (x_q, y_q))$ and v_{pq} be the reigning speed limit on the segment that supports $S_{b,i}$.

- **Case 1:** $d_p \pm d_{pq} = d_q$
  - **If** $d_p - d_{pq} = d_q$ **then** interchange the roles of p and q, now we have $d_p + d_{pq} = d_q$.
  - **If** $d_q \leq t^-_{pq}$ **then** do nothing and move on to the next $S_{b,i}$. See Figure 13 on the left.

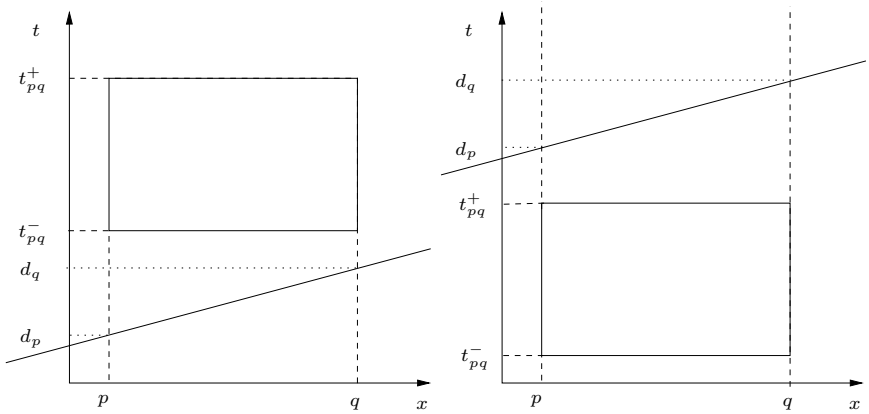

Fig. 13. Case illustration.

- **If** $d_p \geqslant t^+_{pq}$ **then** let replace $\mathcal{S}_{b,r}$ by $\mathcal{S}_{b,r} + \int_0^1 \chi_{pq}$ and move on to the next $S_{b,i}$. See Figure 13 on the right.
- **Else If** $d_p \leq t^-_{pq}$ **then** let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (d_p, x_q, y_q), (d_q, x_q, y_q) \rangle$ **else** let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (t^-_{pq}, x_p, y_p), (t^-_{pq}, x_q, y_q), (d_q, x_q, y_q) \rangle$. Replace $\mathcal{S}_{b,r}$ by $\mathcal{S}_{b,r} + \int \int_{\mathcal{A}} \mu_p \chi_p$. See Figure 14 on the left for the first polygon, and on the right for the second polygon.

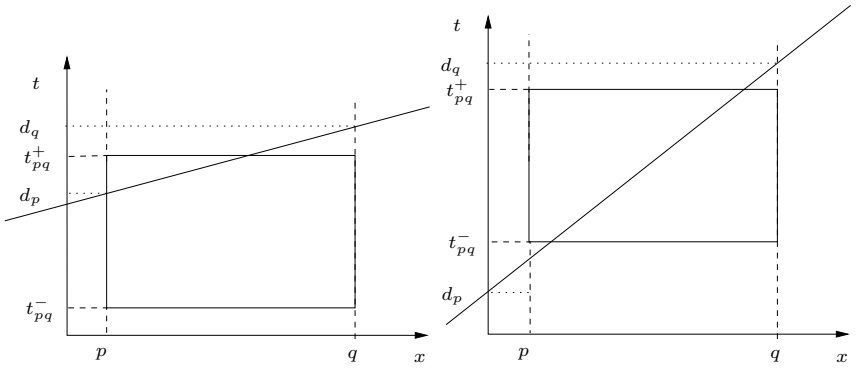

Fig. 14. Case illustration.

Note that although $\mathcal{A}$ is likely to exceed the boundary of $S_{b,i}$, the integral is still well defined because μ_p is zero outside $S_{b,i}$.

– **Case 2:** $d_p \pm d_{pq} \neq d_q$

If this is the case we have to compute the point on the segment where the two fastest paths from this segment to r intersect and distinguish three separate sub-cases. The first is the easiest, when $d_p, d_q \leq t^-_{pq}$ then this inequality will also hold for the time-coordinate where the 2 fastest paths from r intersect, in that case none of the points from $S_{b,i}$ will be able to reach r in time.

- **If** $d_p, d_q \leq t^-_{pq}$ **then** do nothing and proceed to the next sample region. See Figure 15 on the left.

If all time coordinates satisfy $\frac{d_p+d_q-d_{pq}}{2}, d_p, d_q \geqslant t^+_{pq}$ then all points of $S_{b,i}$ will be able to reach r in time. In that case it is pointless to compute intersections and divide the sample region. Hence,

- **If** $\frac{d_p+d_q-d_{pq}}{2}, d_p, d_q \geqslant t^+_{pq}$ **then** replace $\mathcal{S}_r$ by $\mathcal{S}_r + \int_0^1 \chi_{pq}$ and proceed to the next sample region. See Figure 15 on the right.

In the remaining case one of the fastest paths from r intersects $S_{b,i}$ and we need to split $S_{b,i}$ into two regions such that all points in those regions have a unique fastest path to r. We apply the same strategy as in [9]. As shown in Figure 12, we merely need to reduce our computations to the two-dimensional case, compute the x-coordinate of the intersection and multiply that by an appropriate unit vector. In this case $S_{b,i}$ will be split at the spatial

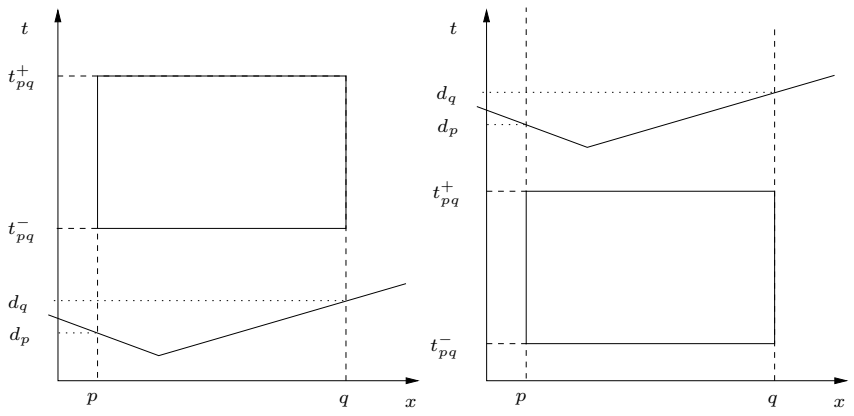

Fig. 15. Case illustration.

point

$$s = (x_s, y_s) = (x_p, y_p) + \left(\frac{d_p - d_q + d_{pq}}{2/v_{pq}}\right) \cdot \frac{(x_q - x_p, y_q - y_p)}{\sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}}.$$

- **Else** Replace $S_{b,i}$ by the two new regions $S'_{b,i}$ and $S''_{b,i}$ where
  - $S'_{b,i} = (p, s, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ f)$ where

$$f(\lambda) = \lambda \cdot \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

  - $S''_{b,i} = (s, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ g)$ where

$$g(\lambda) = \lambda \cdot \sqrt{\frac{(x_q - x_s)^2 + (y_q - y_s)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}} + \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

  and store d_p in the vertex (x_p, y_p), d_q in the vertex (x_q, y_q) and $\frac{d_p + d_q - d_{pq}}{2}$ in the vertex (x_s, y_s) to avoid re-computation in Case 1. Now proceed as in Case 1 for each of these two new regions.

Step 2. *The procedure $S_{e,i}$ is very much like the one outlined in Step 1, except, as indicated in Figure 11, we need to construct the polygons on the other side of the fastest path.*

Let $S_{e,i} = (p, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq})$ where $p = (x_p, y_p), q = (x_q, y_q)$. The following needs to be repeated for each $S_{e,i}$. Let $d_p = t_r + \mathsf{d_{RN}}((x_r, y_r), (x_p, y_p))$, $d_q = t_r + \mathsf{d_{RN}}((x_r, y_r), (x_q, y_q))$, $d_{pq} = \mathsf{d_{RN}}((x_p, y_p), (x_q, y_q))$ and v_{pq} be the reigning speed limit on the segment that supports $S_{e,i}$.

– **Case 1:** $d_p \pm d_{pq} = d_q$

- **If** $d_p - d_{pq} = d_q$ **then** interchange the roles of p and q, now we have $d_p + d_{pq} = d_q$.
- **If** $d_p \geqslant t^+_{pq}$ **then** do nothing and move on to the next $S_{e,i}$.
- **If** $d_q \leq t^-_{pq}$ **then** let replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int_0^1 \chi_{pq}$ and move on to the next $S_{e,i}$.
- **Else If** $d_q \geqslant t^+_{pq}$ **then** let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (d_q, x_p, y_p), (d_q, x_q, y_q) \rangle$ **else** let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (t^+_{pq}, x_p, y_p), (t^+_{pq}, x_q, y_q), (d_q, x_q, y_q) \rangle$. Replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int \int_{\mathcal{A}} \mu_p \chi_p$.

Note that although $\mathcal{A}$ is likely to exceed the boundary of $S_{e,i}$, the integral is still well defined because either μ_p is zero outside $S_{e,i}$.

– **Case 2:** $d_p \pm d_{pq} \neq d_q$

If this is the case we have to compute the point on the segment where the two fastest paths from this segment to r intersect and distinguish three separate sub-cases. The first is the easiest, when $d_p, d_q \leq t^-_{pq}$ then this inequality will also hold for the time-coordinate where the 2 fastest paths from r intersect, in that case none of the points from $S_{b,i}$ will be able to reach r in time.

- **If** $d_p, d_q \geqslant t^+_{pq}$ **then** do nothing and proceed to the next sample region.

If all time coordinates satisfy $\frac{d_p + d_q + d_{pq}}{2}, d_p, d_q \leq t^-_{pq}$ then any moving point starting from r will be able to reach all points in $S_{e,i}$ in time. In that case it is pointless to compute intersections and divide the sample region. Hence,

- **If** $\frac{d_p + d_q + d_{pq}}{2}, d_p, d_q \leq t^+_{pq}$ **then** replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int_0^1 \chi_{pq}$ and proceed to the next sample region.

In the remaining case one of the fastest paths from r intersects $S_{e,i}$ and we need to split $S_{e,i}$ into two regions such that all points in those regions have a unique fastest path to r. We apply the same strategy as in [9]. As illustrated in Figure 12, we merely need to reduce our computations to the two-dimensional case, compute the x-coordinate of the intersection and multiply that by an appropriate unit vector. In this case $S_{e,i}$ will be split at the spatial point

$$s = (x_s, y_s) = (x_p, y_p) + \left(\frac{d_q - d_p + d_{pq}}{2/v_{pq}} \right) \cdot \frac{(x_q - x_p, y_q - y_p)}{\sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}}.$$

- **Else** Replace $S_{e,i}$ by the two new regions $S'_{e,i}$ and $S''_{e,i}$ where
  - $S'_{e,i} = (p, s, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ f)$ where

$$f(\lambda) = \lambda \cdot \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

  - $S''_{e,i} = (s, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ g)$ where

$$g(\lambda) = \lambda \cdot \sqrt{\frac{(x_q - x_s)^2 + (y_q - y_s)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}} + \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

and store d_p in the vertex (x_p, y_p), d_q in the vertex (x_q, y_q) and $\frac{d_p+d_q-d_{pq}}{2}$ in the vertex (x_s, y_s) to avoid re-computation in Case 1. Now proceed as in Case 1 for each of these two new regions.

Output $\mathcal{S}_{b,r}$, $\mathcal{S}_{e,r}$ and $\mathcal{S}_r = \mathcal{S}_{b,r} \cdot \mathcal{S}_{e,r}$.

Now that we have an algorithm for an individual point, we can construct one to visualise this for all points of an uncertain space-time prism envelope. However, we will not do this for all points, we divide the envelope in smaller regions, pick a representative point for each region, compute its probability and assign a suitable colour to that entire region.

The 3D-SPACE-TIME PRISM algorithm outputs a set of polygons. Again we cycle over all the edges of the road network that contain such a polygon. For each such edge we intersect the polygons with a two-dimensional grid on that edge, the size of the grid depends on a pre-chosen resolution. The intersection gives again a set of polygons, where no polygon is larger than the cells of the grid. For each of those we compute the center of mass which we will feed to our POINTPROBABILITY-algorithm. This in turn yields a number, between zero and one, which we associate with that center of mass and its polygon and use it to assign an appropriate colour to the polygon.

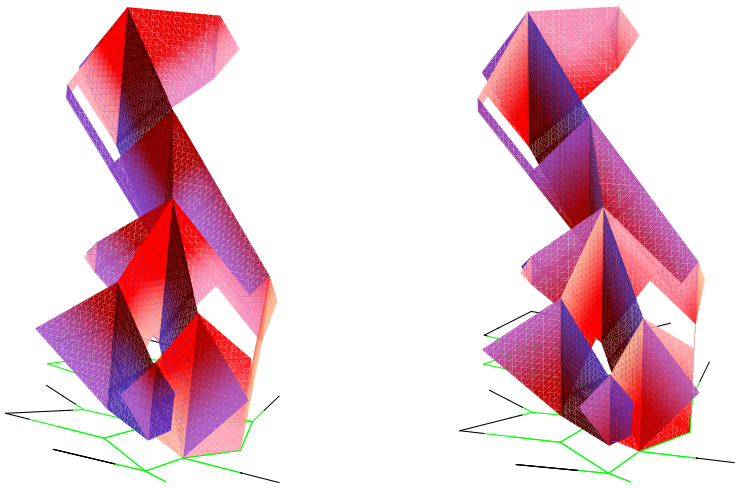

Fig. 16. A space-time prism with the emanating fraction (left) and absorbing fraction (right) coloured in shades of red.

We implemented these algorithms in Mathematica as a proof-of-concept. In Figure 16 on the left we restricted our algorithm to compute the emanating fraction of each spatio-temporal point. In Figure 16 on the right we restricted

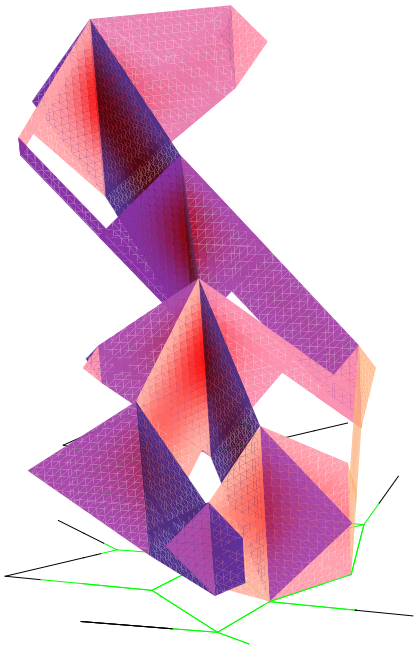

Fig. 17. A space-time prism with the fraction of each space-time point coloured in shades of red.

our algorithm to compute the absorbing fraction of each spatio-temporal point. In Figure 17 our algorithm computed the emanating fraction of each spatio-temporal point with respect to travel from the originating regions to the destination region. In all these examples we assumed a uniform distribution on the sample regions.

5 Applications

In the previous sections we introduced a multitude of new concepts and quantities. Each of those separately and combined can be used in a number of applications which we outline here. Such applications include error analysis and measuring uncertainty and flexibility.

5.1 Measuring flexibility

At the end of the previous section we generated a fully coloured prism where all spatio-temporal points indicated their fraction simultaneously.

Assume that in a point **r** the emanating fraction of **r** with respect to S_b, as defined in Definition 9, is close to one. This means that we can reach **r** from most of S_b. More importantly, this means there exists a path from S_b that reaches the spatial component of **r** and allows us to spend a time at this location that equals almost all of the temporal width of S_b. We illustrate this in Figure 18. The shaded part of S_b covers almost all of S_b, which means the emanating fraction of **r** with respect to S_b is close to one. A fastest path from any point on the bottom of S_b to the spatial location of **r** would be parallel in space-time to the fastest path drawn in Figure 18. Moreover, suppose the temporal height of S_b equals $\triangle t$, then any such path to the right of s from the bottom of S_b arrives at least $\triangle t$ earlier at the spatial location of **r**, we can thus spend at least $\triangle t$-time at the spatial location of **r**. If we leave from the left of s we have a little less than $\triangle t$-time to spend. So, the higher the emanating fraction, the more flexibility we have to choose when and where to leave from S_b and vice versa.

Likewise, assume that in a point **r** the absorbing fraction of **r** with respect to S_e, as defined in Definition 9, is close to one. This means that we can reach most of S_e from **r**. More importantly, we can spend an amount of time, that equals almost all of the temporal width of S_e, at the spatial component **r** before we have to leave again and still be able to reach S_e in time.

This gives us a measure of flexibility we have with respect to the starting and ending regions at each location at each moment in time. A more practical way is to apply a kind of spatial projection on the road network. If we project, for each location, the maximum of these fractions in time onto the road network, we immediately obtain the flexibility we have to reach those locations with respect to our schedule or probable locations to leave from or arrive at.

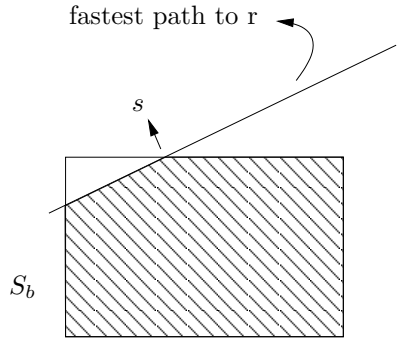

Fig. 18. Illustration of flexibility.

5.2 Measurement errors and space-time prisms

A common strategy in error analysis is to model errors on observed values and analyse how these propagate onto derived values. Using the envelope from Section 3.3, we can do two things.

Firstly, since our sample regions are bounded sets by definition, the envelope is the union of all possible space-time prisms with anchors in the sample regions.

Secondly, as is common in error analysis, we can compute a confidence region on the spatial and temporal component separately of each region, this is again a box-shaped region and compute the envelope with that confidence region as a new sample region. Note that we can not use a simultaneous confidence region for both space and time since those regions are usually elliptical in nature. This again returns the union of all space-time prisms with anchors in those confidence regions.

6 Conclusions and future work

Space-time prisms model uncertainty between sample points, which are usually considered to be perfect measurements. In this paper we extended the notion of sample points to sample regions, which are bounded, sometimes disconnected, subsets of space-time wherein each point is a possible location, with its respective probability, where a moving object could have originated from or arrived in. This model allows us to model measurement errors, multiple possible simultaneous locations and even flexibility of a moving object.

We developed an algorithm that computes the envelope of all space-time prisms that have an anchor in these sample regions and we developed an algorithm that computes for any spatio-temporal point the probability with which a space-time prism, with anchors in these sample regions, contains that point.

We left out a complexity consideration. It is straightforward once you have the number of nodes obtained from the pre-computation, but obtaining that number however is not so trivial. This would rely on defining suitable heuristics that relate the width of temporal intervals to the number of possible distinct edges that can be travelled, which in turn also relies on network density. But these are all aspects we left out in our simplified model and can be tackled in future work.

As for scalability, when treating a chain of prisms, i.e., a lifeline necklace, each prism can be computed separately and completely independent. Our algorithm is thus very scalable on large trajectory samples.

In the simplest case, when all distributions on space and time are uniform, the uncertain prism has a lot of symmetry. Further study is needed to exploit this symmetry and speed up the computation.

In this paper we restricted our sample regions to box shapes, this can easily be extended to other shapes because the intersection we need to compute remains the same, i.e., the intersection of one or two half-spaces with the sample region.

One aspect that has not been studied in this paper is a measure to rank a space-time point's probability inside a single prism. Intuitively one can imagine that points near the edge of the prism are less likely that points on the interior, however, a suitable likelihood function to express this intuition has not been found. A logical step is then to figure out a way to combine both measures and interpret them in a sensible manner.

Acknowledgements

This research has been partially funded by the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery, and by the Research Foundation Flanders (FWO-Vlaanderen), Research Project G.0344.05.

References

1. M. Egenhofer. Approximation of geopatial lifelines. In *SpadaGIS, Workshop on Spatial Data and Geographic Information Systsems*, 2003. Electr. proceedings, 4p.
2. D. Ettema and T. Timmermans. Space-time accessibility under conditions of uncertain travel times: Theory and numerical simulations. *Geographical Analysis*, 39(2):217–240, 2007.
3. T. Hägerstrand. Migration and area: Survey of a sample of swedish migration fields and hypothetical consideration of their genesis. In D. Hannerberg, T. Hägerstrand, and B. Odeving, editors, *Migration in Sweden: A symposium*. The Royal University of Lund, Department of Geography, 1957.
4. T. Hägerstrand. What about people in regional science? *Papers of the Regional Science Association*, 24:7–21, 1970.
5. R.W. Hall. Travel outcome and perfomance: The effect of uncertainty on accessibility. *Transporation Research B*, 17(4):275–290, 1983.
6. Armin Hammand and Amin Hassan Karimi. *Telegeoinformatics: Location-based Computing and Services*. Taylor & Francis, Inc., Bristol, PA, USA, 2004.
7. M.D. Hendricks, M.J. Egenhofer, and Hornsby K. Structuring a wayfinder's dynamic space-time environment. *Lecture Notes in Computer Science*, 2825:75–92, 2003.
8. H.-M. Kim and M.-P. Kwan. Space-time accessibility measures: A geocomputational algorithm with a focus on the feasible opportunity set and possible activity duration. *Journal of Geographical Systems*, 5(1):71–91, 2003.
9. B. Kuijpers and W. Othman. Modelling uncertainty of moving objects on road networks via space-time prisms. *International Journal of Geographical Information Science*.
10. M.-P. Kwan and X.-D. Hong. Network-based constraints-oriented choice set formation using gis. *Journal of Geographical Systems*, 5:139–162, 1998.
11. H.J. Miller. Modeling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems*, 5:287–301, 1991.
12. H.J. Miller. A measurement theory for time geography. *Geographical Analysis*, 37(1):17–45, 2005.

13. H.J. Miller. Place-based versus people-based geographic information science. *Geography Compass*, 1(3):503–535, 2007.
14. H.J. Miller and S.A. Bridwell. A field-based theory for time geography. *Annals of the Association of American Geographers*, 2008.
15. T. Neutens, N. Van de Weghe, F. Witlox, and P. De Maeyer. A three-dimensional network-based space-time prism. *Journal of Geographical Systems*, 10(1):89–107, 2008.
16. T. Neutens, F. Witlox, N. Van de Weghe, and P. De Maeyer. Human interaction spaces under uncertainty. *Transportation Research Record*, 2021:28–35, 2007.
17. D. O'Sullivan, A. Morrison, and J. Shearer. Using desktop gis for the investigation of accessibility by public transport: an isochrone approach. *International Journal of Geographical Information Science*, 14(1):85–104, 2000.
18. D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Advances in Spatial Databases (SSD'99)*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132, 1999.
19. D. Pfoser, N. Tryfona, and Jensen C. Indeterminacy and spatiotemporal data: Basic definitions and case study. *Geoinformatica*, 9(3):211–236, 2005.
20. P. Rietveld. Rounding of arrival and departure times in travel surveys: an interpretation in terms of scheduled activities. *Journal of Transportation and Statistics*, 5(1):71–82, 2002.
21. T. Schwanen. On 'arriving on time', but what is 'on time'? *Geoforum*, 37(6):882–894, 2006.
22. H. Timmermans, T. Arentze, and C.H. Joh. Analysing space-time behaviour: new approaches to old problems. *Progress in human geography*, 26(2):175–190, 2002.
23. G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, 2004.
24. J. Weber and M.-P. Kwan. Bringing time back in: A study on the influence of travel time variations and facility opening hours on individual accessibility. *The Professional Geographer*, 54(2):226–240, 2002.
25. E. Weisstein. Wolfram mathworld. 2007. http://mathworld.wolfram.com/DijkstrasAlgorithm.html.
26. F. Witlox. Evaluating the reliability of reported distance data in urban travel behaviour analysis. *Journal of Transport Geography*, 15(3):172–183, 2007.
27. O. Wolfson. Moving objects information management: The database challenge. In *Proceedings of the 5th Intl. Workshop Next Generation Information Technologies and Systems*, pages 75–89. Springer, 2002.
28. Y.-H. Wu and H.J. Miller. Computational tools for measuring space-time accessibility within transportation networks with dynamic flow. *Journal of Transportation and Statistics*, 4(2/3):1–14, 2002.

Appendix: The MATHEMATICA implementation

```
ClearAll["Global`*"]; $MinPrecision = 20;
Needs["Combinatorica`"]; << Imtek`Polygon`;
$ContextPath; << Imtek`Point`; $ContextPath;
minMaxT[p_] := Module[{i, tmin, tmax}, tmin = Infinity;
    tmax = -Infinity; For[i = 0, i < Length[p], Module[
       {}, If[p[[i]][[3]] < tmin, tmin = p[[i]][[3]]];
       If[p[[i]][[3]] > tmax, tmax = p[[i]][[3]]];],
     i++]; {tmin, tmax}];
counterClockWise2DPolygon[pgon2d_] :=
 Module[{}, If[(pgon2d[[2]][[1]] - pgon2d[[1]][[1]]) *
       (pgon2d[[3]][[2]] - pgon2d[[2]][[2]]) -
      (pgon2d[[2]][[2]] - pgon2d[[1]][[2]]) *
       (pgon2d[[3]][[1]] - pgon2d[[2]][[1]]) < 0,
    Reverse[pgon2d], pgon2d]]
project[g_, r_, pgon_] := Module[{j, gon2d},
  gon2d = {}; For[j = 0, j < Length[pgon],
    Module[{tmp = {g[[2]][[r]][[1]][[1]],
        g[[2]][[r]][[1]][[2]], pgon[[j]][[3]]}},
     gon2d = Append[gon2d, {Norm[tmp - pgon[[j]]],
        pgon[[j]][[3]]}]], j++];
  counterClockWise2DPolygon[gon2d]]
reconstruct3DPolygon[g_, r_, s_, twoDGon_] :=
 Module[{i, ix = g[[2]][[r]][[1]][[1]],
   iy = g[[2]][[r]][[1]][[2]], jx =
    g[[2]][[s]][[1]][[1]], jy = g[[2]][[s]][[1]][[2]],
   afst = Norm[g[[2]][[s]][[1]] - g[[2]][[r]][[1]]],
   ijx, ijy, threeDGon}, ijx = (jx - ix) / afst;
  ijy = (jy - iy) / afst; threeDGon = {};
  For[i = 0, i < Length[twoDGon], threeDGon =
    Append[threeDGon, {ix + twoDGon[[i]][[1]] * ijx,
      iy + twoDGon[[i]][[1]] * ijy,
      twoDGon[[i]][[2]]}], i++]; threeDGon]
included[pol1_, pol2_] := Module[{icheck = True, i}, For[
   i = 0, (i < Length[pol1]) && icheck, icheck = icheck &&
     imsPointInPolygonQ[pol1[[i]], pol2], i++]; icheck]
intersect[g_, r_, s_, p_, vierkantje_] :=
 Module[{p2d = project[g, r, p], ins},
  If[included[vierkantje, p2d], ins = vierkantje,
   ins = imsConvexIntersect[{p2d, vierkantje}, 14.]];
  reconstruct3DPolygon[g, r, s, ins]]

colorPol[g_, Sb_, Se_, r_, s_, weight_, pol_] :=
 Module[{FSb = 0, FSe = 0, totalSbArea = 0, totalSeArea = 0,
   dcr, dcs, center = reconstruct3DPolygon[g, r, s,
       {imsCenterOfMass[project[g, r, pol]]}][[1]],
   i, j}, dcr = Norm[{g[[2]][[r]][[1]][[1]],
       g[[2]][[r]][[1]][[2]], center[[3]]} - center] *
    weight / Norm[g[[2]][[r]][[1]] - g[[2]][[s]][[1]]];
  dcs = Norm[{g[[2]][[s]][[1]][[1]],
       g[[2]][[s]][[1]][[2]], center[[3]]} - center] *
    weight / Norm[g[[2]][[r]][[1]] - g[[2]][[s]][[1]]];
```

```
For[i = 0, i < Length[Sb], totalSbArea =
    totalSbArea + imsArea[{{0, Sb[[i]][[2]][[1]]},
        {Norm[g[[2]][[Sb[[i]][[1]][[1]]]][[1]] -
           g[[2]][[Sb[[i]][[1]][[2]]]][[1]]],
         Sb[[i]][[2]][[1]]},
        {Norm[g[[2]][[Sb[[i]][[1]][[1]]]][[1]] -
           g[[2]][[Sb[[i]][[1]][[2]]]][[1]]], Sb[[i]][[
           2]][[2]]}, {0, Sb[[i]][[2]][[1]]}}], i++];
For[i = 0, i < Length[Se], totalSeArea =
    totalSeArea + imsArea[{{0, Se[[i]][[2]][[1]]},
        {Norm[g[[2]][[Se[[i]][[1]][[1]]]][[1]] -
           g[[2]][[Se[[i]][[1]][[2]]]][[1]]],
         Se[[i]][[2]][[1]]},
        {Norm[g[[2]][[Se[[i]][[1]][[1]]]][[1]] -
           g[[2]][[Se[[i]][[1]][[2]]]][[1]]], Se[[i]][[
           2]][[2]]}, {0, Se[[i]][[2]][[2]]}}], i++];

For[i = 0, i < Length[Sb], Module[
    {dp = center[[3]] - Min[Sb[[i]][[3]][[2]][[r]] + dcr,
        Sb[[i]][[3]][[2]][[s]] + dcs],
     dq = center[[3]] - Min[Sb[[i]][[3]][[3]][[r]] + dcr,
        Sb[[i]][[3]][[3]][[s]] + dcs],
     dpq = Sb[[i]][[3]][[1]], SbPol, split,
     SbWidth = Norm[g[[2]][[Sb[[i]][[1]][[1]]]][[1]] -
        g[[2]][[Sb[[i]][[1]][[2]]]][[1]]], split =
      (dp - dq + dpq) * SbWidth / (2 * Sb[[i]][[3]][[1]]);
    SbPol = {{0, Sb[[i]][[2]][[1]]},
      {SbWidth, Sb[[i]][[2]][[1]]}, {SbWidth,
       Sb[[i]][[2]][[2]]}, {0, Sb[[i]][[2]][[2]]}};
    If[(dp + dpq == dq) || (dp - dpq == dq),
      If[Min[dp, dq] ≥ Sb[[i]][[2]][[2]],
       FSb = FSb + imsArea[SbPol] / totalSbArea,
       If[Min[dp, dq] <= Sb[[i]][[2]][[1]],
        FSb = FSb + imsArea[imsConvexIntersect[{SbPol, {{0,
                 Max[dp, dq]}, {0, Min[dp, dq]}, {SbWidth,
                 Min[dp, dq]}}}, 14.]] / totalSbArea,
        FSb = FSb + imsArea[imsConvexIntersect[
              {SbPol, {{0, dp}, {0, Sb[[i]][[2]][[1]]},
                {SbWidth, Sb[[i]][[2]][[1]]},
                {SbWidth, dq}}}, 14.]] / totalSbArea]],
     If[((dp + dq - dpq) / 2) ≥ Sb[[i]][[2]][[2]],
      FSb = FSb + imsArea[SbPol] / totalSbArea,
      If[((dp + dq - dpq) / 2) <= Sb[[i]][[2]][[1]],
       FSb = FSb + (imsArea[imsConvexIntersect[
              {SbPol, {{0, dp}, {0, (dp + dq - dpq) / 2},
                {split, (dp + dq - dpq) / 2}}}, 14.]] +
            imsArea[imsConvexIntersect[{SbPol,
              {{split, (dp + dq - dpq) / 2}, {SbWidth,
                (dp + dq - dpq) / 2}, {SbWidth,
                dq}}}, 14.]]) / totalSbArea,
       FSb = FSb + (imsArea[imsConvexIntersect[
              {SbPol, {{0, dp}, {0, Sb[[i]][[2]][[1]]},
                {split, Sb[[i]][[2]][[1]]}, {split,
                 (dp + dq - dpq) / 2}}}, 14.]] +
            imsArea[imsConvexIntersect[{SbPol,
              {{split, (dp + dq - dpq) / 2}, {split,
                Sb[[i]][[2]][[1]]}, {SbWidth,
                Sb[[i]][[2]][[1]]}, {SbWidth, dq}}},
             14.]]) / totalSbArea]]]], i++];
```

```
For[i = 0, i < Length[Se], Module[
    {dp = center[[3]] + Min[Se[[i]][[3]][[2]][[r]] + dcr,
        Se[[i]][[3]][[2]][[s]] + dcs],
     dq = center[[3]] + Min[Se[[i]][[3]][[3]][[r]] + dcr,
        Se[[i]][[3]][[3]][[s]] + dcs],
     dpq = Se[[i]][[3]][[1]], SePol, split,
     SeWidth = Norm[g[[2]][[Se[[i]][[1]][[1]]]][[1]] -
        g[[2]][[Se[[i]][[1]][[2]]]][[1]]], split =
      (dq - dp + dpq) * Norm[g[[2]][[Se[[i]][[1]][[1]]]][[
          1]] - g[[2]][[Se[[i]][[1]][[2]]]][[1]]] /
        (2 * Se[[i]][[3]][[1]]);
    SePol = {{0, Se[[i]][[2]][[1]]},
      {SeWidth, Se[[i]][[2]][[1]]}, {SeWidth,
       Se[[i]][[2]][[2]]}, {0, Se[[i]][[2]][[2]]}};
    If[(dp + dpq == dq) || (dp - dpq == dq),
     If[Max[dp, dq] <= Se[[i]][[2]][[1]],
      FSe = FSe + imsArea[SePol] / totalSeArea,
      If[Max[dp, dq] >= Se[[i]][[2]][[2]], FSe =
        FSe + imsArea[imsConvexIntersect[{SePol, {{0,
                Max[dp, dq]}, {0, Min[dp, dq]}, {SeWidth,
                Max[dp, dq]}}}, 14.]] / totalSeArea,
       FSe = FSe + imsArea[imsConvexIntersect[
             {SePol, {{0, Se[[i]][[2]][[2]]}, {0, dp},
               {SeWidth, dq}, {SeWidth, Se[[i]][[2]][[
                 2]]}}}, 14.]] / totalSeArea]],
     If[((dp + dq + dpq) / 2) <= Se[[i]][[2]][[1]],
      FSe = FSe + imsArea[SePol] / totalSeArea,
      If[((dp + dq + dpq) / 2) >= Se[[i]][[2]][[2]],
       FSe = FSe + (imsArea[imsConvexIntersect[
               {SePol, {{0, (dp + dq + dpq) / 2}, {0, dp},
                 {split, (dp + dq + dpq) / 2}}}, 14.]] +
             imsArea[imsConvexIntersect[{SePol,
                {{split, (dp + dq + dpq) / 2}, {SeWidth, dq},
                 {SeWidth, (dp + dq + dpq) / 2}}},
               14.]]) / totalSeArea,
       FSe = FSe + (imsArea[imsConvexIntersect[
               {SePol, {{0, Se[[i]][[2]][[2]]},
                 {0, dp}, {split, (dp + dq + dpq) / 2},
                 {split, Se[[i]][[2]][[2]]}}}, 14.]] +
             imsArea[imsConvexIntersect[{SePol,
                {{split, Se[[i]][[2]][[2]]},
                 {split, (dp + dq + dpq) / 2}, {SeWidth, dq},
                 {SeWidth, Se[[i]][[2]][[2]]}}},
               14.]]) / totalSeArea]]], i++];

 {pol, FSb, FSe, FSb * FSe}]

computeFragmentedPolygon[g_, Sb_, Se_,
  r_, s_, weight_, pgons3d_, resolution_] :=
 Module[{j, k, l, h, m, n, tmin = Infinity,
   tmax = -Infinity, fragPol = {}, coloredPol = {}},
  For[j = 0, j < Length[pgons3d], Module[{ttmin, ttmax},
    {ttmin, ttmax} = minMaxT[pgons3d[[j]]];
    If[ttmin < tmin, tmin = ttmin];
    If[ttmax > tmax, tmax = ttmax];], j++];
```

```
For[h = 0, h < Length[pgons3d], Module[
    {p = pgons3d[[h]]}, m = (tmax - tmin) / resolution;
    n = Norm[g[[2]][[r]][[1]] - g[[2]][[s]][[1]]] /
      resolution; For[k = 0, k <= m, For[l = 0, l <= n,
      Module[{vierkantje = {{0 + (l - 1) * resolution,
           tmin + (k - 1) * resolution}, {l * resolution,
           tmin + (k - 1) * resolution}, {l * resolution,
           tmin + k * resolution}, {0 + (l - 1) * resolution,
           tmin + k * resolution}}, instemp},
       instemp = intersect[g, r, s, p, vierkantje];
       If[! MemberQ[fragPol, instemp] &&
         (Length[instemp] > 2),
        Module[{}, fragPol = Append[fragPol, instemp];
         coloredPol = Append[coloredPol, colorPol[
            g, Sb, Se, r, s, weight, instemp]];]];],
      l++], k++];], h++]; coloredPol];


drawBead[g_, speedlimits_, resolution_] :=
 Module[{arrivals, departures, validprism,
   startingregiontimes, destinationregiontimes,
   i, bead3d = {}, bead2d = {}, rn = {}, Vbead = {},
   weights = g[[1]], distances = g[[1]], speeds = g[[1]],
   numberofvertices = Length[g[[2]]],
   numberofedges = Length[g[[1]]]},
  g = SetEdgeWeights[g, WeightingFunction → Euclidean];
  For[i = 0, i < numberofedges,
   Module[{}, distances[[i]] = g[[1]][[i]][[2]][[2]];
    speeds[[i]] = speedlimits[[
      Random[Integer, {1, Length[speedlimits]}]]];
    weights[[i]] = distances[[i]] / speeds[[i]];], i++];
  g = SetEdgeWeights[g, weights];
  (*Initializing the arrivals and departures arrays*)
  startingregiontimes = Sb;
  destinationregiontimes = Se;
  arrivals = g[[2]];
  departures = g[[2]];
  For[i = 0, i < Length[arrivals],
   arrivals[[i]] = Infinity, i++];
  For[i = 0, i < Length[departures],
   departures[[i]] = Infinity, i++];
  (*Computing the earliest possible
   arrival time for each node*)
  For[i = 0, i < Length[Sb],
   Module[{r = Sb[[i]][[1]][[1]], s = Sb[[i]][[1]][[2]],
     j, tb = Sb[[i]][[2]][[1]], tarrivals1 =
      Dijkstra[g, {Sb[[i]][[1]][[1]]}][[1]][[2]],
     tarrivals2 = Dijkstra[g, {Sb[[i]][[1]][[2]]}][[
        1]][[2]]}, For[j = 0, j < Length[tarrivals1],
     Module[{}, Sb = ReplacePart[Sb, {i, 3, 2, j} →
         tarrivals1[[j]]]; Sb = ReplacePart[Sb,
        {i, 3, 3, j} → tarrivals2[[j]]]], j++];
    For[j = 0, j < Length[arrivals], arrivals[[j]] =
      tb + Min[arrivals[[j]] - tb, tarrivals1[[j]],
        tarrivals2[[j]]], j++];], i++];
  validprism = True;
```

```
(*Computing the latest possible
 departure time for each node*)
For[i = 0, i < Length[Se],
  Module[{r = Se[[i]][[1]][[1]], s = Se[[i]][[1]][[2]],
    j, te = Se[[i]][[2]][[2]], tdepartures1 =
     Dijkstra[g, {Se[[i]][[1]][[1]]}][[1]][[2]],
    tdepartures2 =
     Dijkstra[g, {Se[[i]][[1]][[2]]}][[1]][[2]]},
   For[j = 0, j < Length[tdepartures1], Module[{}, Se =
      ReplacePart[Se, {i, 3, 2, j} -> tdepartures1[[j]]];
     Se = ReplacePart[Se, {i, 3, 3, j} ->
        tdepartures2[[j]]]], j++];
   For[j = 0, j < Length[departures], departures[[j]] =
     te - Min[departures[[j]] - te, tdepartures1[[j]],
       tdepartures2[[j]]], j++];];
(*Verifying if EVERY point in the destination
 region CAN be reached*)
Module[{drs}, For[i = 0,
   (i < Length[Se]) && validprism, Module[
    {j, t = Se[[i]][[2]][[1]], r = Se[[i]][[1]][[1]],
     s = Se[[i]][[1]][[2]], found = False},
    For[j = 0, (j < numberofedges) && (! found),
     If[Se[[i]][[1]] == g[[1]][[j]][[1]],
      Module[{}, found = True; drs = weights[[j]];
       Se = ReplacePart[Se, {i, 3, 1} -> drs];
       destinationregiontimes[[i]] =
        (departures[[r]] + departures[[s]] - drs) / 2;]],
     j++]; validprism = validprism && ((arrivals[[r]] +
          arrivals[[s]] + drs) / 2 <= t)], i++]];
(*Verifying if EVERY point in the starting
 region CAN be started from*)
Module[{drs}, For[i = 0,
   (i < Length[Sb]) && validprism, Module[
    {j, t = Sb[[i]][[2]][[2]], r = Sb[[i]][[1]][[1]],
     s = Sb[[i]][[1]][[2]], found = False},
    For[j = 0, (j < numberofedges) && (! found),
     If[Sb[[i]][[1]] == g[[1]][[j]][[1]],
      Module[{}, found = True; drs = weights[[j]];
       Sb = ReplacePart[Sb, {i, 3, 1} -> drs];
       startingregiontimes[[i]] =
        (arrivals[[r]] + arrivals[[s]] + drs) / 2;]], j++];
    validprism = validprism && ((departures[[r]] +
          departures[[s]] - drs) / 2 >= t)], i++]];
If[validprism, Print["prism is valid"],
  Print["prism is NOT valid"]];
For[i = 0, i < numberofvertices,
  If[departures[[i]] ≥ arrivals[[i]],
   Vbead = Vbead ⋃ {i}], i++];
```

```
If[validprism, For[i = 0, i < numberofedges,
  Module[{v = speeds[[i]], L = distances[[i]],
    w = weights[[i]], r = g[[1]][[i]][[1]][[1]],
    s = g[[1]][[i]][[1]][[2]], pol3d = {}, rx, ry,
    sx, sy, rsx, rsy}, rx = g[[2]][[r]][[1]][[1]];
   ry = g[[2]][[r]][[1]][[2]]; sx =
    g[[2]][[s]][[1]][[1]]; sy = g[[2]][[s]][[1]][[2]];
   rsx = (sx - rx) / L; rsy = (sy - ry) / L;
   If[¬ (MemberQ[Vbead, r] ⋁ MemberQ[Vbead, s]),
    rn = rn ⋃ {{{rx, ry, 0}, {sx, sy, 0}}}];
   If[(MemberQ[Vbead, r]) ⋀ (MemberQ[Vbead, s]),
    If[((arrivals[[r]] + w) ≤ departures[[s]]) ⋁
      ((arrivals[[s]] + w) ≤ departures[[r]]),
     bead2d = bead2d ⋃ {{{rx, ry, 0}, {sx, sy, 0}}},
     Module[{rd = v * (departures[[r]] - arrivals[[r]]) / 2,
       sd = v * (departures[[s]] - arrivals[[s]]) / 2},
      bead2d = bead2d ⋃ {{{rx, ry, 0}, {rx + rd * rsx,
           ry + rd * rsy, 0}}}; bead2d = bead2d ⋃
        {{{sx, sy, 0}, {sx - sd * rsx, sy - sd * rsy, 0}}};
      rn = rn ⋃ {{{rx + rd * rsx, ry + rd * rsy, 0},
          {sx - sd * rsx, sy - sd * rsy, 0}}};]]];
   If[(MemberQ[Vbead, r]) ⋀ (¬ MemberQ[Vbead, s]),
    Module[{rd = v * (departures[[r]] - arrivals[[r]]) / 2},
     bead2d = bead2d ⋃ {{{rx, ry, 0},
         {rx + rd * rsx, ry + rd * rsy, 0}}}; rn = rn ⋃
       {{{rx + rd * rsx, ry + rd * rsy, 0}, {sx, sy, 0}}};]];
   If[(MemberQ[Vbead, s]) ⋀ (¬ MemberQ[Vbead, r]),
    Module[{sd = v * (departures[[s]] - arrivals[[s]]) / 2},
     bead2d = bead2d ⋃ {{{sx, sy, 0},
         {sx - sd * rsx, sy - sd * rsy, 0}}}; rn = rn ⋃
       {{{rx, ry, 0}, {sx - sd * rsx, sy - sd * rsy, 0}}};]];
   If[MemberQ[Vbead, r], If[
     v * (departures[[r]] - arrivals[[r]]) / 2 ≤ L,
     pol3d = pol3d ⋃ {{{rx, ry, arrivals[[r]]},
         {rx, ry, departures[[r]]}, {rx + rsx * v *
            (departures[[r]] - arrivals[[r]]) / 2, ry +
           rsy * v * (departures[[r]] - arrivals[[r]]) / 2,
          (departures[[r]] + arrivals[[r]]) / 2}}},
     pol3d = pol3d ⋃ {{{rx, ry, arrivals[[r]]},
         {rx, ry, departures[[r]]},
         {sx, sy, departures[[r]] - w}, {sx, sy,
          arrivals[[r]] + w}}}]]; If[MemberQ[Vbead, s],
    If[v * (departures[[s]] - arrivals[[s]]) / 2 ≤ L,
     pol3d = pol3d ⋃ {{{sx, sy, arrivals[[s]]},
         {sx, sy, departures[[s]]}, {sx - rsx * v *
            (departures[[s]] - arrivals[[s]]) / 2, sy -
           rsy * v * (departures[[s]] - arrivals[[s]]) / 2,
          (departures[[s]] + arrivals[[s]]) / 2}}},
     pol3d = pol3d ⋃ {{{sx, sy, arrivals[[s]]},
         {sx, sy, departures[[s]]},
         {rx, ry, departures[[s]] - w},
         {rx, ry, arrivals[[s]] + w}}}]];
   If[(MemberQ[Vbead, r]) ⋀ (MemberQ[Vbead, s]),
    Module[{},
     If[(arrivals[[r]] + L / v) < departures[[s]],
      pol3d = pol3d ⋃ {{{sx, sy, arrivals[[r]] + w},
          {sx, sy, departures[[s]]},
          {rx, ry, departures[[s]] - w},
          {rx, ry, arrivals[[r]]}}}];
     If[(arrivals[[s]] + w) < departures[[r]],
      pol3d = pol3d ⋃ {{{sx, sy, arrivals[[s]]}, {rx, ry,
          arrivals[[s]] + w}, {rx, ry, departures[[r]]},
         {sx, sy, departures[[r]] - w}}}]]];
```

```
(*Add the corrected starting regions to the prism*)
     If[validprism, Module[{nfound, j},
        nfound = True;
        For[j = 0, (j < Length[Sb]) && nfound,
         If[(r == Sb[[j]][[1]][[1]]) &&
           (s == Sb[[j]][[1]][[2]]),
          Module[{}, nfound = False;
           pol3d = pol3d ⋃ {{{g[[2]][[r]][[1]][[1]],
                 g[[2]][[r]][[1]][[2]], Sb[[j]][[2]][[
                  1]]}, {(g[[2]][[r]][[1]][[1]] +
                    g[[2]][[s]][[1]][[1]]) / 2,
                 (g[[2]][[r]][[1]][[2]] + g[[2]][[s]][[
                      1]][[2]]) / 2, startingregiontimes[[
                  j]]}, {g[[2]][[s]][[1]][[1]],
                 g[[2]][[s]][[1]][[2]],
                 Sb[[j]][[2]][[1]]}}};]], j++];
        (*Add the corrected destination
         regions to the prism*)
        nfound = True;
        For[j = 0, (j < Length[Se]) && nfound,
         If[(r == Se[[j]][[1]][[1]]) &&
           (s == Se[[j]][[1]][[2]]),
          Module[{}, nfound = False;
           pol3d = pol3d ⋃ {{{g[[2]][[r]][[1]][[1]],
                 g[[2]][[r]][[1]][[2]], Se[[j]][[2]][[
                  2]]}, {g[[2]][[s]][[1]][[1]],
                 g[[2]][[s]][[1]][[2]], Se[[j]][[2]][[
                  2]]}, {(g[[2]][[r]][[1]][[1]] +
                    g[[2]][[s]][[1]][[1]]) /
                  2, (g[[2]][[r]][[1]][[2]] +
                    g[[2]][[s]][[1]][[2]]) / 2,
                 destinationregiontimes[[
                  j]]}}}]], j++]]];
     bead3d = bead3d ⋃ computeFragmentedPolygon[g,
        Sb, Se, r, s, w, pol3d, resolution]], i++]];
 If[validprism, {Show[Graphics3D[
     {Table[{EdgeForm[], RGBColor[1,
         1 - bead3d[[i]][[2]], 1 - bead3d[[i]][[2]]],
        Opacity[1], Polygon[bead3d[[i]][[1]]]},
       {i, 1, Length[bead3d]}],
      Table[{RGBColor[0, 1, 0], Line[bead2d[[i]]]},
       {i, 1, Length[bead2d]}],
      Table[{RGBColor[0, 0, 0], Line[rn[[i]]]},
       {i, 1, Length[rn]}]},
     Boxed → False, PlotRange → All]],
   Show[Graphics3D[{Table[{EdgeForm[], RGBColor[1,
         1 - bead3d[[i]][[3]], 1 - bead3d[[i]][[3]]],
        Opacity[1], Polygon[bead3d[[i]][[1]]]},
       {i, 1, Length[bead3d]}],
      Table[{RGBColor[0, 1, 0], Line[bead2d[[i]]]},
       {i, 1, Length[bead2d]}],
      Table[{RGBColor[0, 0, 0], Line[rn[[i]]]},
       {i, 1, Length[rn]}]},
     Boxed → False, PlotRange → All]],
   Show[Graphics3D[{Table[{EdgeForm[], RGBColor[1,
         1 - bead3d[[i]][[4]], 1 - bead3d[[i]][[4]]],
        Opacity[1], Polygon[bead3d[[i]][[1]]]},
       {i, 1, Length[bead3d]}], Table[{RGBColor[0, 1, 0],
        Line[bead2d[[i]]]}, {i, 1, Length[bead2d]}],
      Table[{RGBColor[0, 0, 0], Line[rn[[i]]]},
       {i, 1, Length[rn]}]},
     Boxed → False, PlotRange → All]}]]
```

The code above can be reused. The following are the parameters with which the above code was called. There is a set of speed limits, which are assigned randomly to each edge of the road network every time the above code is called. Notice that S_b consists of two regions and S_e of one.

```
speedlimits = {.85, 1, 1.35}; resolution = .05;
g = Graph[{{{1, 2}}, {{2, 5}}, {{5, 6}},
    {{6, 13}}, {{13, 14}}, {{14, 19}}, {{18, 19}},
    {{14, 17}}, {{15, 17}}, {{16, 17}}, {{15, 18}},
    {{6, 7}}, {{3, 6}}, {{8, 13}}, {{8, 20}}, {{4, 8}},
    {{4, 5}}, {{5, 21}}, {{2, 9}}, {{7, 10}}, {{11, 14}},
    {{6, 8}}, {{3, 10}}, {{2, 3}}, {{7, 23}}, {{11, 13}}},
   {{{-1.4101562, -0.08417988}},
    {{-1.2695312, 0.08418}},
    {{-0.76953125, 0.44412196}},
    {{-0.7578125, -0.74020314}},
    {{-0.98828125, -0.2351234}},
    {{-0.4453125, 0.03773582}},
    {{-0.26953125, 0.6473149}},
    {{-0.18359375, -0.89114666}},
    {{-1.0507812, 0.86211896}},
    {{-0.484375, 1.3265603}}, {{0.62890625, 1.0478954}},
    {{0.421875, -0.5253992}},
    {{0.125, 0.08418}}, {{0.8984375, 0.56023216}},
    {{1.296875, -0.27576208}},
    {{1.2226562, -0.9608128}}, {{1.0, -0.30478954}},
    {{1.6445312, 0.47895503}}, {{1.1992188, 1.0072569}},
    {{0.40625, -1.1930335}}, {{-1.25, -1.030479}},
    {{0.7734375, -0.43831635}},
    {{0.15234375, 1.018868}}}];
Sb = {{{4, 5}, {0, 0.5}, {0, g[[2]], g[[2]]}},
   {{2, 3}, {0.25, .85}, {0, g[[2]], g[[2]]}}};
Se = {{{14, 17}, {3, 3.15}, {0, g[[2]], g[[2]]}}};
drawBead[g, speedlimits, resolution]
```