Received: 17 December 2020 | Revised: 8 July 2021 | Accepted: 9 July 2021

DOI: 10.1111/tgis.12821

RESEARCH ARTICLE

Transactions in GIS WILEY

Multimodal travel-time maps with formally correct and schematic isochrones

Axel Forsch | Youness Dehbi | Benjamin Niedermann | Johannes Oehrlein | Peter Rottmann | Jan-Henrik Haunert

Institute of Geodesy and Geoinformation, Working Group Geoinformation, University of Bonn, Bonn, Germany

Correspondence
University of Bonn, Meckenheimer Allee 172, Bonn 53115, Germany.
Email: forsch@igg.uni-bonn.de

Funding Information
This research was supported by the German Research Foundation (DFG), grant 5451/7-1, within the DFG priority program, grant 1894

Abstract

The automatic generation of travel-time maps is a prerequisite for many fields of application such as tourist assistance and spatial decision support systems, for example to analyze the accessibility of health and social facilities. The task is to determine outlines of zones that are reachable from a user's location in a given amount of time. In this work we focus on travel-time maps with a formally guaranteed SEPARATION PROPERTY in the sense that a zone exactly contains the part of the road network that is reachable within a pre-defined time from a given starting point and start time. In contrast to other automated methods that create travel-time maps, our approach generates schematized travel-time maps that reduce the visual complexity by representing each zone by an octilinear polygon, that is, the edges of the polygons use only eight pre-defined orientations. We aim for octilinear polygons with a small number of bends to further optimize the legibility of the map. The reachable parts of the road network are determined by the integration of timetable information for different modes of public transportation, for example buses, trains or ferries, and pedestrian walkways based on a multimodal time-expanded network. Moreover, the travel-time maps generated visualize multiple travel times using a map overlay of different time zones and taking

natural barriers such as rivers into account. In experiments on real-world data we compare our schematic visualizations to travel-time maps created with other visualization techniques with respect to simple but robust quality measures such as the number of bends and the perimeter of the zones.

1 | INTRODUCTION

Travel-time maps provide the user with an easily comprehensible visualization of areas that are reachable from a selected location within a prescribed amount of time. The boundaries of such areas are commonly referred to as *isochrones*. We present a new method for generating *schematic* travel-time maps automatically – a preview of some results generated with our method is provided in Figure 1. The motivation for using a schematic visualization is to avoid the high graphical complexity that can occur with existing solutions, which we review in detail in Section 2. However, we aim not only for a low graphical complexity but also for a visualization inducing a formally correct classification of the transport network into reachable and unreachable parts.

Before giving a formal definition of the property ensured by our method, we discuss a classical approach (in the following referred to as *time buffering*) and its possible flaws. An early map of Melbourne generated with time buffering is shown in Figure 2. The approach consists of two steps.

1. Travel times are estimated from a selected point to all stations in the transport network (in the map of Melbourne the tram and railway network).
2. To visualize the area reachable within some amount of time τ, a disk is drawn around each reachable station s such that the disk's radius equals the available amount of time $\tau - \tau_s$ (where τ_s is the travel time to s) multiplied by a constant (which may correspond to the assumed walking speed outside the transport network).

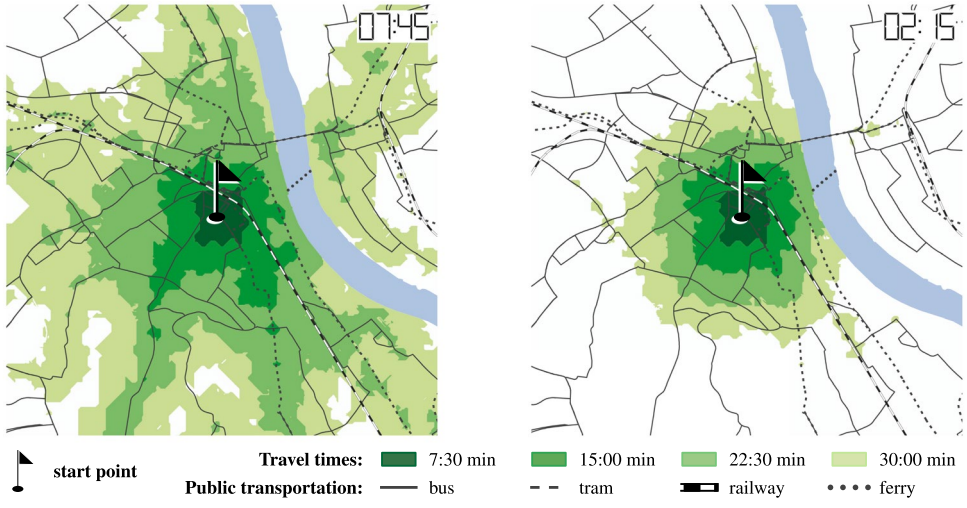


FIGURE 1 Travel-time maps generated for the city of Bonn, Germany. The maps show the accessibility of the road network starting from the same location for different start times in the morning (left) and during the night (right) of a work day

Usually, as in the example of Figure 2, this approach is repeated with multiple different values for τ, resulting in multiple areas (in the following referred to as *time zones*). Furthermore, the time-buffering approach can easily be generalized to situations where in Step 1 accurate travel times are estimated not only for a discrete set of stations but also for the continuous set of all points in a geometric graph – an example we generated ourselves using public transportation timetable data and road data is shown in Figure 3a. We observe that many points in the network that in Step 1 were classified as unreachable (dashed lines) fall into the area highlighted as reachable (red region). We

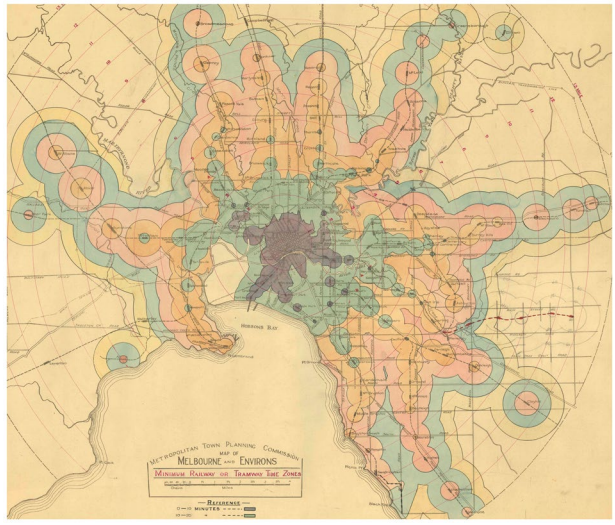


FIGURE 2 An early travel-time map of Melbourne rail transport travel times, 1910–1922. The public transportation is taken into account: for each reachable station the remaining travel time is visualized by concentric circles. *Source*: Melbourne and Metropolitan Tramways Board—State Library of Victoria

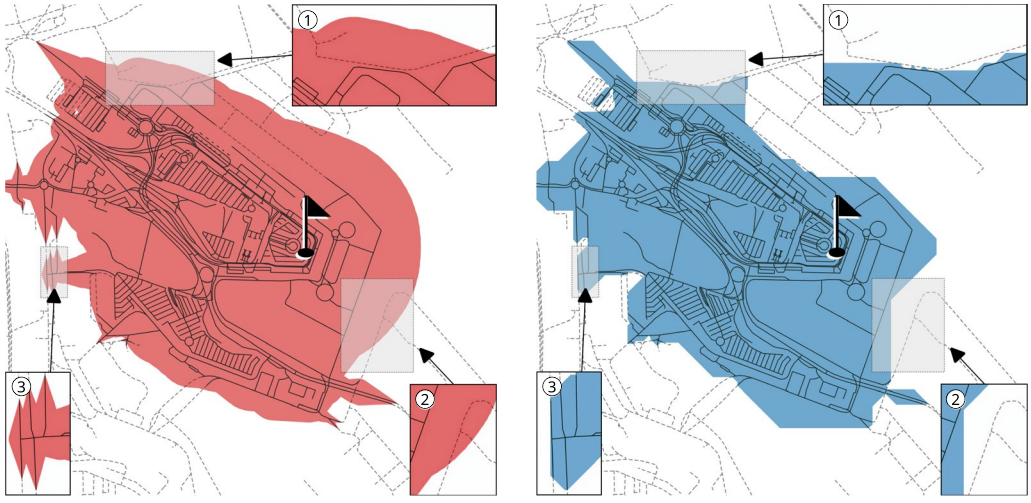


FIGURE 3 A single time zone of 15 minutes starting at the flag. Unreachable roads are dashed. (a) Time zone created by buffering the remaining travel time for each point of the road network. ①–② The time zone covers unreachable roads. ③ Time zone has ramified outgrowths. (b) Time zone created by our approach. ①–② Time zone only covers reachable roads. ③ SCHEMATIZATION simplifies the visualization

consider this inappropriate, since there is no evidence that the empty space between the roads is traversable. More precisely, the assumed constant walking speed for areas outside the network lacks a good justification. We do not make this assumption and keep the classification of points in the network as it results from Step 1. Still, we would like to use areas to visualize the reachable part of the network. We argue that schematic polygons are particularly well suited for this, because they keep the graphical complexity low. The travel-time map created with our approach is shown in Figure 3b. Aside from the LOW VISUAL COMPLEXITY, another advantage of schematized time zones is that SCHEMATIZATION is widely applied for the visualizations of transport networks, such as metro maps. Since schematic metro maps are so commonly applied in practice, it can be assumed that most people understand that the geometric information provided with schematic polygons needs to be taken with a grain of salt (i.e., not all points of the map plane contained in a time zone we display are actually reachable) yet qualitative information displayed for the transport network, such as the classification into reachable and unreachable parts, is correct.

We formalize the correct classification of the transport network into reachable and unreachable parts as follows. A location t in a road network $\mathscr{R}$ is *reachable* from the user's location s in time τ if there is a route r from s to t using footpaths in $\mathscr{R}$ and connections in the public transportation network such that it takes at most time τ to travel along r; all other locations of $\mathscr{R}$ are *unreachable*. Due to the possibly disconnected structure of the reachable subnetwork, we describe its outline, the time zone of temporal extent τ, not by a single polygon, but rather by a set of polygons whose boundaries do not intersect; we explicitly allow holes in the polygons modeling unreachable parts contained in reachable parts of the network. We consider a time zone *formally correct* if it satisfies the following SEPARATION PROPERTY.

Definition 1 (SEPARATION PROPERTY). The time zone Z satisfies the *SEPARATION PROPERTY* if each reachable location of $\mathscr{R}$ is contained in one of the polygons of Z and each unreachable location of $\mathscr{R}$ is not.

In contrast to previous work (Baum, Bläsius, Gemsa, Rutter, & Wegner, 2018; Gamper, Böhlen, Cometti, & Innerebner, 2011; Krismer, Specht, & Gamper, 2017), we follow the idea of *schematic mapping* to use only octilinear orientations for the edges of the time zone, that is, they are either horizontal, vertical or diagonal. To that end, we discretize the solution space by an octilinear grid. In the exceptional case that our algorithm does not find a strictly octilinear polygon that separates the reachable and unreachable nodes, we allow non-octilinear lines. To keep the visual complexity low, we heuristically minimize the number of bends of the polygons.

Altogether, the core contribution of this article is an algorithm for creating a high-quality, schematized travel-time zone for a starting location given a start time and maximum travel times. It fulfills the following requirements:

1. *MULTIMODAL NETWORKS*. The travel-time map is based on a multimodal transportation network taking public transport and pedestrian movement into account.
2. *LOW VISUAL COMPLEXITY*. The travel-time map has LOW VISUAL COMPLEXITY without ramified structures.
3. *SEPARATION PROPERTY*. The travel-time map is formally correct and represents reachable and unreachable locations in the road network accurately.
4. *SCHEMATIZATION*. The travel-time map is schematized such that the shape of each of its time zones is restricted to a fixed number of edge directions.

We have developed our method particularly for users roaming through a city, for example, tourists exploring different sightseeing spots distributed all over the city. In that scenario, typically the user gets around on foot and uses public transportation to bridge larger distances. Hence, to determine the reachable part of the road network in Step 1, we consider both public transportation and all parts of the road network that are accessible on foot. Taking public transportation into account poses additional challenges when creating travel-time maps. Firstly, when determining the reachable part of the road network in Step 1, the underlying routing component needs to support integrated pathfinding in the road and public transportation network. To that end, we utilize the time-expanded model introduced

FIGURE 4 Incorporating natural barriers. (a) Time zones without cutting natural barriers. (b) Time zones after cutting natural barrier

by Pyrga, Schulz, Wagner, and Zaroliagis (2008) for routing in public transportation networks with timetables, and connect it with the road network. This enables us to compute realistic routes whose travel times depend on the current timetable. Secondly, a time zone is not necessarily a single component, but may consist of multiple components each possibly containing holes representing unreachable parts. This needs to be taken into account in Step 2.

Additionally we show how to embed this algorithm in a holistic framework for creating travel-time maps with multiple, correctly nested travel-time zones. To that end, we introduce the following extensions.

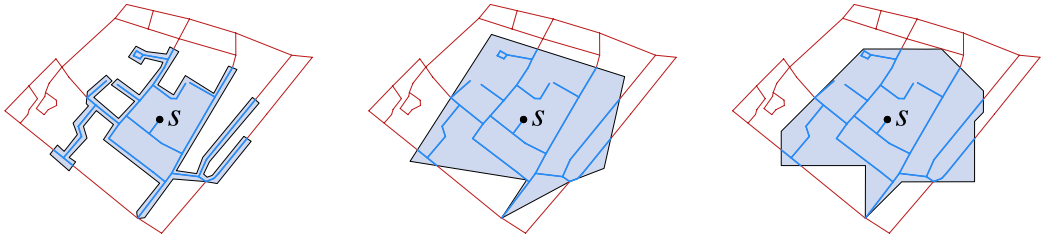
1. *Multiple travel times*. Time zones of different travel times are overlaid. To avoid cluttered maps, the algorithm is adapted such that the time zones are correctly nested.
2. *Clipping*. A post-processing step incorporates natural barriers such as rivers that cannot be overcome on foot. Overlaying them with time zones may lead to the wrong impression that a region is easily accessible on foot; see Figure 4a. We therefore clip time zones to natural barriers; see Figure 4b.
3. *Polishing*. To enhance the visualization a *morphological closing operation* borrowed from image processing is adapted on graph structures and applied to resolve small artifacts.
4. *Multiple zoom levels*. In a generalization step the time zone is adapted to multiple zoom levels showing different levels of detail.

The article is organized as follows. In Section 2 we discuss related work. In Section 3 we give a high-level overview of the components of our approach, which we explain in greater detail in Sections 4–6. In Section 7 we describe possible extensions. In Section 8 we present our experiments on real-world data and their evaluation. Section 9 concludes. The source code can be downloaded at https://www.geoinfo.uni-bonn.de/travel-time-maps.

2 | RELATED WORK

Two basic algorithmic problems lie at the core of the process of creating travel-time maps. The first is about determining the reachable part of the transportation network. The result is a possibly disconnected set of components of the network that describe the reachable subnetwork for a given travel time. For our scenario, it needs to fulfill the requirement of supporting MULTIMODAL NETWORKS. The second algorithmic problem is then about the visualization of the obtained components. This problem is related to our requirements of LOW VISUAL COMPLEXITY, SEPARATION PROPERTY and SCHEMATIZATION.

For computing the reachable part of the transportation network, previous work has considered various types of networks and routing algorithms that were specially engineered for solving this task in real-time applications. Bauer, Gamper, Loperfido, Profanter, Putzer, and Timko (2008) presented a simple routing algorithm that takes both a road network as well as a bus network into account. Gamper et al. (2011) proposed a formal



(a) outline of network (b) minimal bends (c) schematic

FIGURE 5 Sketches of different methods for creating travel-time maps. (a) Outline of network. (b) Minimal bends. (c) Schematic

definition which describes an isochrone as the (possibly disconnected) subgraph of a transportation network that can be reached from a given starting point in a specific time. Based on Dijkstra's algorithm for finding shortest paths in graphs, they presented an algorithm for computing isochrones in MULTIMODAL NETWORKS. Gamper, Böhlen and Innerebner (2012) and Krismer et al. (2017) improved that algorithm with respect to its running time and memory consumption. Moreover, Krismer, Specht and Gamper (2014) considered the computation of multiple isochrones for different travel times but the same starting point. Baum, Buchhold, Dibbelt, and Wagner (2019) adapted customized route planning to computing isochrones in dynamic road networks. They specially engineered their algorithms to deploy them in interactive scenarios taking large networks (i.e., spanning entire continents) into account.

When it comes to the visualization of isochrones less research has been conducted so far. O'Sullivan, Morrison and Shearer (2000) proposed an approach deployed in a geoinformation system for visualizing isochrones that take public transportation into account. They suggested computing the remaining walking time for each station and visualizing the possible pedestrian movement either by concentric circles around the stations or more complex regions modeling the area that is accessible on foot. These regions are joined and inaccessible parts are cut out. However, their approach does not enforce the SEPARATION PROPERTY as the concentric circles can overlap unreachable areas. Krajzewicz and Heinrichs (2016) split the map into cells and computed for each cell the required travel time for a given starting point and multimodal transportation. They used this information to aggregate and color the cells with respect to the required travel time. However, based on the cells this easily yields structures with branched outgrowths in the visualization. Marciuska and Gamper (2010) presented two approaches for visualizing isochrones in road networks. The first creates a buffer with user-specific size around each reachable edge in the road network, and the second (see Figure 5a) constructs a single polygon enclosing the reachable part which is then enlarged by buffering. Neither approach enforces the SEPARATION PROPERTY, but when choosing a small buffer size the time zones only contain a small number of unreachable roads. Baum et al. (2018) presented an approach for visualizing isochrones in road networks with a strong focus on algorithm engineering. They represented the isochrones by polygons that minimize the number of bends and satisfy the SEPARATION PROPERTY; see Figure 5b. Their approach consists of the following steps:

1. Determine the reachable and unreachable part of the road network.
2. Planarize the road network resolving bridges and tunnels.
3. Determine the faces in the planarization that separate the reachable parts from the unreachable part. These faces are merged to one face separating reachable and unreachable regions.
4. For each reachable region compute a polygon that encloses that region and separates the unreachable from the reachable part. All of these polygons combined form the time zone.

In our approach we apply similar steps in a slightly different order. We planarize the road network only once and reuse this information for multiple queries. Further, we consider multimodal transportation and represent the

TABLE 1 Overview of different methods for creating isochrones towards fulfilling our requirements

	MULTIMODAL NETWORKS	LOW VISUAL COMPLEXITY	SEPARATION PROPERTY	SCHEMATIZATION
Melbourne (1910)	No	No	No	circles
O'Sullivan et al. (2000)	Yes	No	No	No
Bauer et al. (2008)	Yes	–	Yes	–
Marciuska and Gamper (2010)	No	No	No	No
Krajzewicz and Heinrichs (2016)	Yes	No	No	No
Baum et al. (2018)	No	min. bends	Yes	No
Our approach	Yes	min. bends	Yes	octilinear

isochrones by schematized polygons; see Figure 5c. An overview of related work for creating travel-time maps is given in Table 1 by categorizing the methods towards fulfilling the requirements introduced in Section 1.

SCHEMATIZATION has also been investigated in other contexts. Most prominently, SCHEMATIZATION is used to represent transit networks such as metro systems. The interested reader is referred to surveys on automated layout methods for transit maps by Nöllenburg (2014) and Niedermann, Takahashi, and Nöllenburg (2019), Wu, Niedermann, Takahashi, Roberts, and Nöllenburg (2020). Moreover, Buchin, Meulemans, Renssen, and Speckmann (2016) studied the automated simplification and SCHEMATIZATION of territorial outlines preserving area and topology constraints. To capture the shape of arbitrary polygons, Bouts, Kostitsyna, van Kreveld, Meulemans, Sonke, and Verbeek (2016) presented a mapping to simple grid polygons based on Hausdorff or Fréchet distance. Recently, Bonerath, Niedermann and Haunert (2019) presented an approach for visualizing points annotated with time-stamps based on schematized, octilinear α-shapes. In order to construct the octilinear polygons we create a shortcut graph which represents all candidate edges of the time zone. Similar graph structures are used for line simplification (Imai & Iri, 1988) and the simplification of footprints of buildings (Haunert & Wolff, 2010).

Empirical evaluations of LOW VISUAL COMPLEXITY have been conducted in the context of graph drawings. Purchase (2000) identify the number of bends and the number of crossings in a graph drawing as the most significant factors for the usability of graph drawings. Further research on edge crossings showed that not only should the number of crossings be minimized but also their crossing angle should be maximized (Huang, Hong, & Eades, 2008). Ware, Purchase, Colpoys, and McGill (2002) highlight the importance of path continuity for shortest-path perception. We incorporate these findings into our approach by minimizing the number of bends in our time zones and by using octilinear polygons, which have a fixed smallest opening angle of $22.5°$, thus improving path continuity.

We provide travel-time maps at different zoom levels selecting the most important roads in small-scale maps based on their attributes. Other methods for selective omission in road networks can be applied. Brewer, Stanislawski, Buttenfield, Sparks, McGilloway, and Howard (2013) presented an automated approach for thinning of road networks by removing features. For network generalization Chimani, van Dijk and Haunert (2014) introduced an approach that successively deletes the edges preserving the connectivity of the graph. Zhou and Li (2016) proposed a method for the empirical determination of geometric parameters to decide which roads should be retained or eliminated at a specific scale.

3 | WORKFLOW

We present a workflow for creating travel-time maps that consists of three components, which can be enhanced by three extensions; see Figure 6. The input is public transportation data consisting of timetable information and station coordinates, a road network, an origin within the road network, the start time of the query and a set T of travel times.

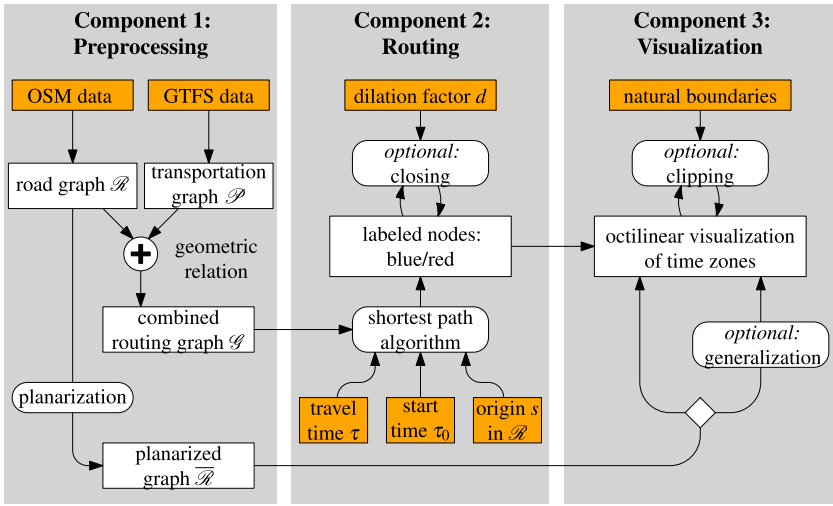


FIGURE 6 Illustration of workflow. The preprocessing component creates different graph structures that are used in the routing component and the visualization component. The routing component marks all reachable nodes blue and all unreachable nodes red. The visualization component creates a time zone for each travel time

3.1 | Preprocessing component

The input data are prepared by preprocessing graph structures that can be reused for multiple travel-time maps with different origins. Hence, we execute this component only once in advance, investing some computation time to accelerate queries made by the user. In our experiments we used General Transit Feed Specification (GTFS) data for public transportation and OpenStreetMap (www.openstreetmap.org, OSM) data for modeling the road network; other data sources may also be integrated. Applying the time-expanded model by Pyrga et al. (2008), we build a graph $\mathcal{P}$ modeling the public transportation. Further, we represent the road network as a geometric graph $\mathcal{R}$. We merge both graphs into one graph $\mathcal{G}$ that is used in the routing component for routing and determining all reachable nodes of $\mathcal{R}$. Further, we *planarize* $\mathcal{R}$ to obtain a graph $\overline{\mathcal{R}}$ whose embedding is planar, that is, there are no crossings between two edges. The planarization is explained in detail in Section 4. We only use $\overline{\mathcal{R}}$ in the visualization component for the purpose of computing the visualization polygons, but not for routing in the routing component. The details are found in Section 4.

3.2 | Routing component

Using a shortest path algorithm on $\mathcal{G}$, we compute the locations in $\mathcal{R}$ that are reachable from a given starting point s and start time τ_0 within travel time $\tau \in T$. We subdivide each edge e that is only partly reachable from s by an additional node at the point with time distance τ from s. Hence, afterwards the reachability of $\mathcal{R}$ corresponds to a coloring of the edges: all reachable edges are *blue* and all unreachable edges are *red*. The details are found in Section 5.

3.3 | Visualization component

This component creates the time zones of the travel-time map. It expects the planarized road network $\overline{\mathcal{R}}$ and for each travel time $\tau \in T$ the edge coloring created by the previous component. For each travel time it creates a set of

schematic polygons containing exactly all reachable edges in $\overline{\mathcal{R}}$. In the case of multiple travel times, the approach considers the creation of the time zones in an integrated way. The details are found in Section 6.

3.4 | Extensions

The workflow can be enhanced by three optional extensions. These aim to further simplify the shape of the created time zones by removing small holes and taking into account natural boundaries and zoom levels. The details are found in Section 7.

4 | PREPROCESSING COMPONENT

In this section we explain how the input data are preprocessed to obtain graph structures that we use in the routing and visualization Components.

4.1 | Road network

We model the road network as a geometric graph $\mathcal{R} = (V_{\mathcal{R}}, E_{\mathcal{R}})$, that is, each node corresponds to a point and each edge corresponds to a line segment whose end points are the incident nodes of the edge. We assume that the course of a road between two junctions is described by a path consisting of degree-2 vertices. In order to accelerate routing we can contract degree-2 vertices, but for the visualization component we also need to consider these nodes to maintain the shape of the network. As we consider pedestrians navigating through the city in a combination of public transportation and walking we only include paths that are accessible by pedestrians; in particular, we exclude all highways and speedways. Further, we annotate each edge $e \in E_{\mathcal{R}}$ with the time that is necessary to traverse it; in our experiments we assumed a constant walking speed of 5 km/h. Note, however, that we make no assumption outside the road network.

Due to tunnels and bridges the given embedding of the road graph $\mathcal{R}$ is not necessarily planar, that is, there may be edges that cross each other. However, for the visualization component we do not need this additional information, but we can make the embedding planar by subdividing the edges with additional vertices at their intersections as similarly done in Baum et al. (2018). The result is a plane graph $\overline{\mathcal{R}}$, which we utilize for creating the time zones in the visualization component. We note that for routing, we use the road graph $\mathcal{R}$ still containing all non-planarized bridges and tunnels.

4.2 | Public transportation network

We assume that the timetable for the public transportation network is given. It is described by a set S of *stations* and a set Γ of *trips*. Each trip $t \in \Gamma$ is a sequence of stations that are served by a vehicle in chronological order. Hence, we describe t as a sequence of *stops* such that each stop is defined by a station $\sigma \in S$ and two times, namely the arrival time and the departure time of the bus or train. Based on this timetable we construct a graph $\mathcal{P} = (\mathcal{V}_{\mathcal{P}}, \mathcal{E}_{\mathcal{P}})$ following the time-expanded model for timetable information presented by Pyrga et al. (2008). As we introduce some problem-specific adaptions we describe this model for the convenience of the reader; see also Figure 7.

For each stop h of each trip we introduce two nodes u and v that represent the arrival and departure of the means of transportation at a certain station σ, respectively. We call u the *arrival node* and v the *departure node* of the stop, and annotate u with its *arrival time* τ_u and v with its *departure time* τ_v. Further, we insert a directed edge

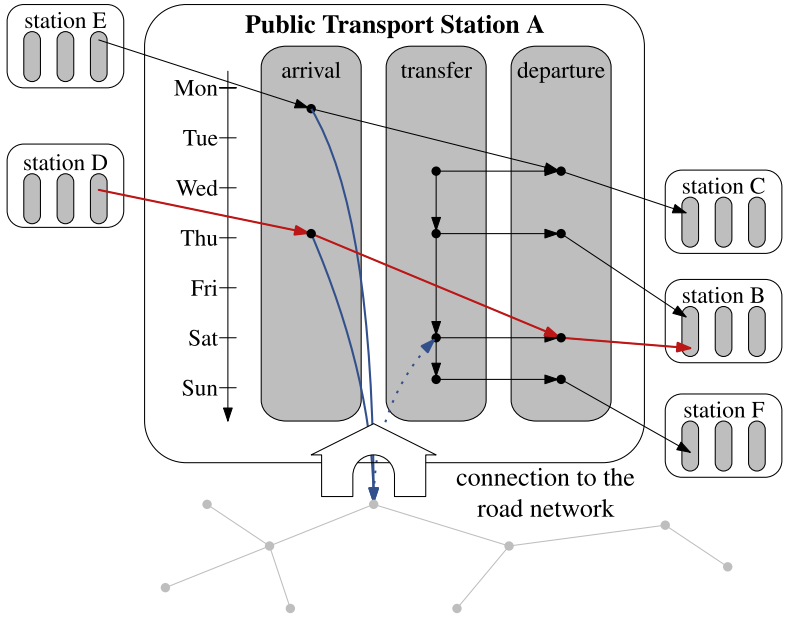


FIGURE 7 Time-expanded model for an example station *A*. Four trips occur at the station: two starting in station *A* and two passing through. The change of a connection is modeled by transfer nodes. The connection to the road network is marked in blue

from *u* to *v*. For each trip, each pair of consecutive stops h, h' are concatenated by adding an edge from the departure node of *h* to the arrival node of h'.

In order to model transfers from one trip to another, we introduce a *transfer nodew* for each stop *h* of each trip. We annotate *w* with a *transfer time* τ_w that is equal to the departure time of *h*. We introduce a directed edge from *w* to *v*, modeling the access to a trip. Finally, we connect the transfer nodes of the same station with each other. To that end, let H_σ be the set of all stops at a station σ. We sort H_σ in increasing order with respect to their transfer times. We introduce directed edges between each pair of consecutive transfers in H_σ. These edges model necessary waiting times at σ.

Further, each edge is annotated with the time difference between its target node and its source node. This difference models the cost for traveling along this edge. The model provides us with the possibility of finding the best journey through the public transportation network, for example with Dijkstra's algorithm. Further, it supports minimal transfer times and a cyclic repetition of the timetable; for more details see Pyrga et al. (2008).

4.3 | Joining road graph and public transportation graph

For the routing component we join the road graph $\mathcal{R}$ and the public transportation graph $\mathcal{P}$ into one graph $\mathcal{G}$ as follows. For each station σ we use its geographic location to identify the closest node *u* in $\mathcal{R}$; we call *u* an *access point* of the public transportation system. To keep the experiments simple we merely used the geographic locations of the stations, but more sophisticated approaches based on the entrances of the stations could be used to make the routing more accurate. To model the user leaving the public transportation system, we connect each arrival node *v* of σ with the access point *u* via a directed edge from *v* to *u*. As each trip has a pre-defined departure time at each station, we cannot simply insert edges between *u* and all transfer nodes of σ modeling the user entering the public transportation system. Instead we do this on demand when computing a shortest journey in $\mathcal{G}$ in the following component.

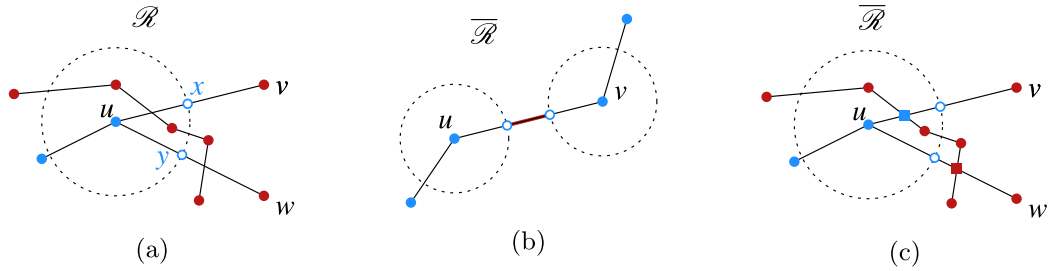


FIGURE 8 The reachability is modeled by subdividing edges. (a) The node x (y) subdivides the edge (u, v) (u, w) at the last reachable position. (b) Special case: the nodes u and v are reachable from different directions. (c) The coloring of $\mathcal{R}$ is transferred to $\overline{\mathcal{R}}$ and the nodes (squares) that are introduced for the planarization are colored

5 | ROUTING COMPONENT

To determine all reachable nodes in the road network we make use of the graph $\mathcal{G}$ as defined in the previous component. For a starting point s in the road network of $\mathcal{G}$ we use Dijkstra's algorithm to determine all nodes that are reachable in time τ. From a technical point of view, the algorithm determines for each node u the smallest journey time that is necessary to reach u. Thus, when routing in the part of $\mathcal{G}$ representing the road network we determine for each node u the earliest time at which we arrive assuming that we are given a start time for the journey (e.g. Monday 2 p.m.). We call this the *earliest arrival time* at u. Further, each time the algorithm reaches an access point u of a station σ in the public transportation system, it temporarily introduces an edge to the transfer node of σ whose transfer time suits the earliest arrival time of u best; see the dotted blue arc in Figure 7. With this we obtain for each node of $\mathcal{G}$ its earliest arrival time. Using a standard termination criterion, we let Dijkstra's algorithm only consider nodes that are actually reachable in time τ such that we do not consider the entire network. We mark all reachable nodes blue and all others red. Further, for each edge e that is incident to both a blue and a rednode we determine the location on e that is still reachable from s in time τ; see Figure 8a. At this location we subdivide e by a blue *dummy node*. A special case occurs when the sum of the remaining distances at two adjacent, reachable nodes u and v is smaller than the length of edge $e = \{u, v\}$, see Figure 8b. In this case e is subdivided by two additional dummy nodes with the middle part being colored red. We further transfer the coloring and the newly inserted nodes into the planarized graph $\overline{\mathcal{R}}$, additionally coloring the nodes that have been introduced to planarize $\mathcal{R}$; see Figure 8c. Each such node becomes blue if it subdivides an edge e in $\mathcal{R}$ that is only incident to blue nodes; otherwise it becomes red. Altogether, we obtain a coloring of $\overline{\mathcal{R}}$ defining all nodes either blue or red. Due to the insertion of the dummy nodes, this node coloring induces an edge coloring: edges that are incident to two reachable nodes are also reachable and all other edges are unreachable.

6 | VISUALIZATION COMPONENT

In this section we describe how to compute the time zones of the travel-time map based on the coloring of the previous step (Section 5). We first explain this for a single travel time and then for multiple travel times.

6.1 | Single Travel Time

In this subsection we assume a single travel time. Thus, we are given the planarized road network $\overline{\mathcal{R}}$ and a coloring such that each edge and each node are either blue (reachable) or red(unreachable). Further, we assume that the

outer face only consists of rednodes but no blue nodes; we can always insert an unreachable bounding box and connect it to the road network.

We first observe that $\overline{\mathcal{R}}$ has faces that have both redand blue edges; see Figure 9. We call an edge that is incident to both a blue and a rednode a *gate*. Further, we call the reachable node of a gate its *port*. Removing the gates from $\overline{\mathcal{R}}$ decomposes the graph into a set of components $C_1, \ldots, C_\ell$ such that each component is either completely blueor red. We call these the *colored components* of $\overline{\mathcal{R}}$.

6.1.1 | Base case

We first consider the case that $C_2, \ldots, C_\ell$ have the same color, but C_1 has the opposite color; see Figure 10a. We describe a procedure that creates a single octilinear polygon such that C_1 lies inside and $C_2, \ldots, C_\ell$ lie outside the polygon. Without loss of generality we assume that C_1 is blue and the other components are red. Let $v_1, \ldots, v_k$ be the ports of $\overline{\mathcal{R}}$ on the boundary of C_1 in the order in which we encounter them when going along the boundary of C_1; we choose an arbitrary starting point and define $v_{k+1} := v_1$. We observe that any two consecutive ports v_i, v_{i+1} are incident to the same face of $\overline{\mathcal{R}}$, which we denote by f_i.

The base case consists of three steps; see Figure 10. In the first step we create for each pair v_i, v_{i+1} of consecutive ports an octilinear grid G_i contained in f_i. In the second step, we fuse these grids to one large grid G. In the final step, we use G to determine an octilinear polygon by finding an optimal path through G.

Step 1. For each pair v_i, v_{i+1} of consecutive ports with $1 \le i \le k$ we first compute an octilinear grid G_i that is contained in f_i and connects v_i and v_{i+1}; see Figure 11. To that end, we shoot from both ports octilinear rays and compute the line-segment arrangement $\mathcal{L}$ of these rays restricted to the face f_i; see Figure 11a. If $\mathcal{L}$ forms one component, we use it as grid G_i. Otherwise, we refine $\mathcal{L}$ as follows. We uniformly subdivide the bounding box of f_i by further nodes from which we shoot additional octilinear rays; see Figure 11b. We insert these into $\mathcal{L}$, restricting them to f_i. We call the number of nodes on the bounding box the *degree of refinement d*. We double the degree of refinement until $\mathcal{L}$ is connected or a threshold $d_{\max}$ is exceeded; see Figure 11c. In the latter case we also insert the boundary of f_i into $\mathcal{L}$ to guarantee that $\mathcal{L}$ is connected. Later on, when creating the octilinear polygon we only use the boundary edges of f_i if necessary.

Step 2. In the following we interpret each grid G_i as a geometric graph such that the grid points are the nodes of the graph and the segments connecting the grid points are the edges of the graph. We union these graphs into one large graph G. More precisely, G is the graph that contains all nodes and edges of the grids $G_1, \ldots, G_k$. In particular, each port v_i is represented by two nodes x_i and y_i in G such that x_i stems from G_{i-1} and y_i stems from G_i; for $i = 1$ we define $x_1 = v_k$. We connect two grids G_{i-1} and G_i in G by introducing the directed edge (x_i, y_i) in G for $2 \le i \le k$.

Step 3. In the following let $s := y_1$ and $t := x_1$. We compute a path P from s to t in G such that P has minimum number of bends, that is, there is no other path from s to t that has fewer bends; see Figure 12b. To that end, we use Dijkstra's algorithm on the linear dual graph of G, allowing us to penalize bends in the cost function. If that the

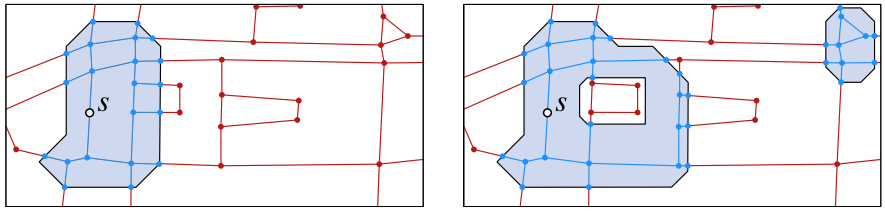


FIGURE 9 Reachabilities for a single travel time. (a) The blue nodes and edges form a connected component. (b) The blue and rednodes and edges form multiple connected components

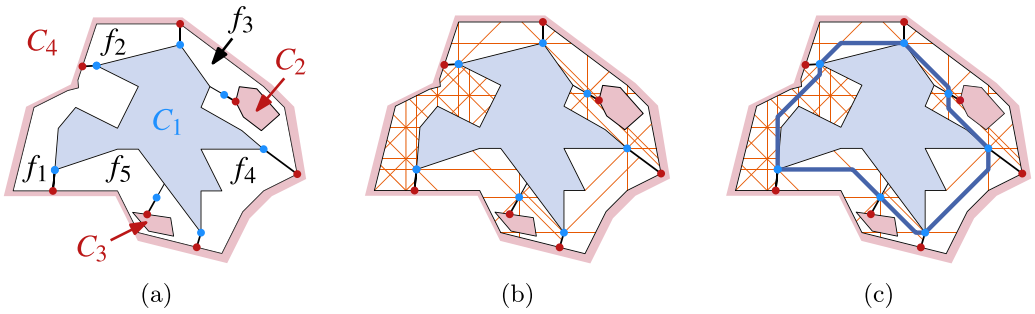


FIGURE 10 Creating an octilinear polygon that encloses the component C_1 (blue) of all reachable nodes and edges. (a) The faces $f_1, \ldots, f_5$ surround the component C_1. (b) Each face is subdivided by an octilinear grid (Step 1). Furthermore, these grids are connected to one large grid G that is split by the port between f_1 and f_5 (Step 2). (c) An octilinear polygon is constructed by computing a bend-minimal path through G

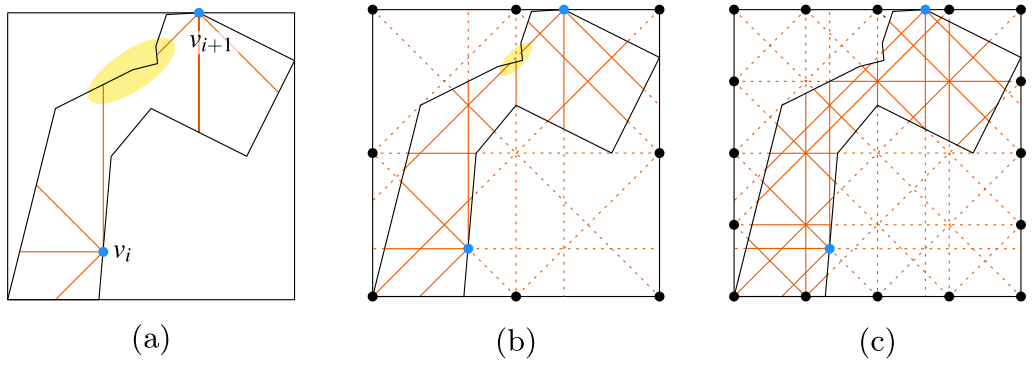


FIGURE 11 The face f_1 of the example shown in Figure 10 is subdivided by octilinear rays based on vertices on the bounding box. (a) Shooting octilinear rays from v_i and v_{i+1} does not yield a connected line arrangement (see yellow highlight). (b)–(c) The bounding box of f_1 is successively refined by vertices shooting octilinear rays until they connect v_i and v_{i+1}

choice of P is not unique, because there are multiple paths with the same number of bends, we use the geometric length of the path as a tie-breaker, preferring paths that are shorter. As s and t have the same geometric location, the path P forms an octilinear polygon O. We note that by using directed edges at the ports, we guarantee that P crosses the port so that no unreachable component can be enclosed by O; see Figure 12b.

Only if in the second step we could not find a connected octilinear grid for a face do we relax the requirement that P is octilinear. In that case P might also contain edges of the respective face. In order to minimize the number of these possibly non-octilinear edges, we penalize them with high costs when searching for P. We note that O is not necessarily simple, as we calculate each segment between two ports individually. This can lead to two segments of the same face crossing each other; see Figure 12c. In this exceptional case we transform O into a simple polygon with holes by joining the overlapping parts. We note that this can change the number of bends in the polygon's outline. We consider this appropriate as we do not search for a global optimum solution for the number of bends, but only the optimal solution in our model.

6.1.2 | General case

If $\overline{\mathcal{R}}$ has multiple reachable and multiple unreachable colored components, we enclose the components in multiple polygons. To that end, we assign to each component a level that states the degree to which it is nested into

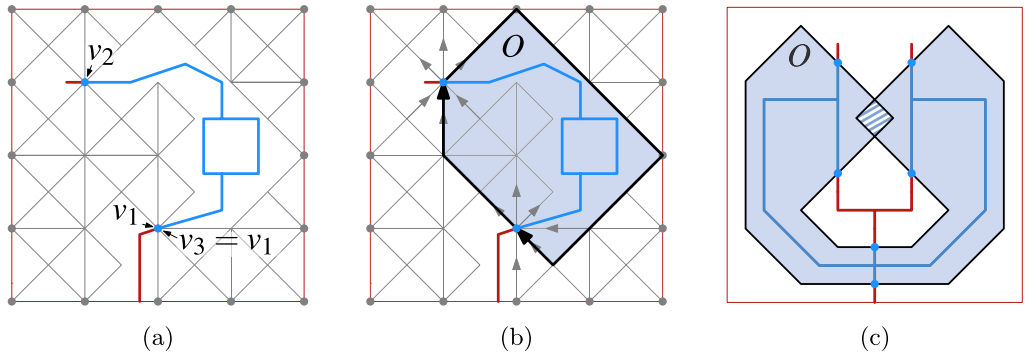


FIGURE 12 Routing step. (a)–(b) The polygon O must not contain unreachable parts. For the routing step the edges incident to ports are directed accordingly. (c) The path O is not necessarily simple

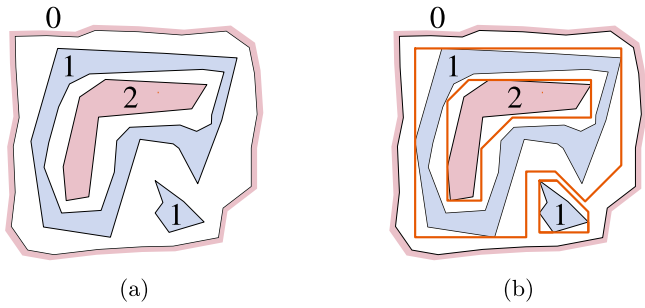


FIGURE 13 The reachable (blue) and unreachable (red) parts of the road graph consist of multiple components. (a) Each component has a level. (b) Each internal component is enclosed by a schematized polygon

other components; see Figure 13a. The idea is that blue components of odd level and redcomponents of even level are enclosed by time zones. For the assignment of the levels, consider the graph $\overline{\mathcal{R}}'$ that we obtain from $\overline{\mathcal{R}}$ by removing its gates. A component that lies in the outer face of $\overline{\mathcal{R}}'$ has level 0. A component C that lies in the internal face of another component C' with level i gets assigned to level $i + 1$. For each blue component with even level and each red component with odd level a time zone is computed respectively; see Figure 13b. Hence, we obtain a time zone such that blue and red edges and nodes of $\overline{\mathcal{R}}$ are separated. Further, time zones of different levels are either disjoint or nested. However, time zones of the same level might partly intersect. We resolve this by simply computing the union of all time zones enclosing components of the same level. Then the resulting set of time zones is this union.

6.2 | Multiple travel times

For multiple travel times we compute the time zones of the travel times in decreasing order. However, we cannot just simply combine them, as routing through the octilinear grids of the different travel times may result in time zones that partly intersect. We observe that the edges and nodes of $\overline{\mathcal{R}}$ are correctly separated for each travel time, but the map may contain unnecessary overlaps between time zones. We therefore proceed as follows. Let $\tau_1, \ldots, \tau_\ell$ be the travel times in increasing order, that is, $\tau_1 < \cdots < \tau_\ell$. For τ_ℓ we compute the set $\mathcal{Z}_\ell$ of time zones as described in Section 6.1. For τ_i with $i < \ell$ we slightly adapt the procedure of Section 6.1. When computing the grid graph G of a single colored component we restrict the graph to the union of the time zones contained in $\mathcal{Z}_{i+1}$. More specifically, we intersect its geometric embedding with the union of the time zones in $\mathcal{Z}_{i+1}$. The result is a smaller grid graph G' that we use for routing

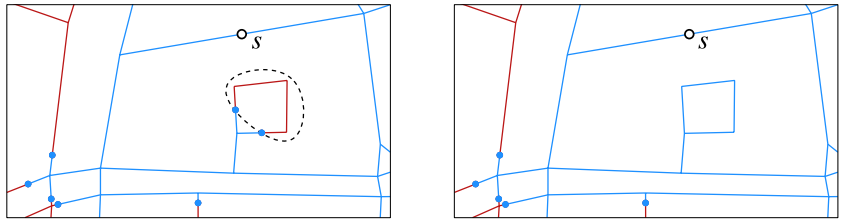


FIGURE 14 Closing operation. (a) A small group of unreachable (red) nodes in the bluearea. (b) After the closing operation the hole is filled

and computing the time zone enclosing the corresponding component. Altogether, we obtain a set of time zones for all travel times. We draw them in the order as their area decreases and color each time zone with a different color.

7 | EXTENSIONS

In this section we briefly describe three extensions that we use to enhance the visualization. The first one can be plugged into the routing component and the second and third ones can be plugged into the visualization component.

7.1 | Closing

Initial experiments showed that the routing component typically returns a colored graph that consists of a few large colored components and a variety of small colored components. For example, due to the choice of the travel time it may happen that a few nodes and edges are unreachable while all others nearby are reachable; see Figure 14. These small unreachable *islands* produce unpleasant-looking "ruptures" in the time zones. To improve the visualization the closing extension removes these holes. A time zone created with this extension thus does not strictly fulfill the SEPARATION PROPERTY any more! Nonetheless, we deem it reasonable to fill these small holes, as the time zones depend on the assumption that the user has a walking speed of exactly 5 km/h, which introduces an exaggerated accuracy. To that end, we utilize the technique of *morphological closing* applied in image processing to suppress small patches of noise (Serra, 1983). The idea is to first extend the area of interest, which removes the holes (dilation) and then to reduce the area to restore the original outer boundaries (erosion).

We adapt this method for our problem as follows. For dilation, we mark every node that is reachable in time τ in blue; see also Figure 14. Then we use an increased time $\tau_d = \tau \cdot (1 + d)$ with $d > 0$ to calculate a buffer region, that is, we compute all nodes that are reachable in time τ_d and mark all nodes in *yellow* that are not already colored blue; we call d the *dilation factor* of the closing operation. All remaining uncolored nodes become red. For erosion, we determine for every yellow node that is adjacent to a red node the yellow component that contains that node. We color all nodes in that component red. Thus, the original boundaries of the blue components are restored, while the nodes in small holes of the blue components remain yellow. Finally, we color the remaining yellow nodes blue, which closes the holes. The greater the chosen value of d, the larger is the hole that is closed by this operation.

7.2 | Travel-time maps at different zoom levels

Depending on the concrete zoom level, digital maps typically show different levels of detail. For example, while in large-scale maps all roads are shown, in small-scale maps only the most important roads are displayed to the user. We

FIGURE 15 Generalization of time zones for different zoom levels. (a) Small streets yield an octilinear polygon with many bends. (b) Minor roads are ignored resulting in a SCHEMATIZATION with fewer bends

implement this concept in travel-time maps as follows. While for large-scale maps we aim for an outline of the time zone that clearly separates the reachable from the unreachable part of the road network, for small-scale maps we deem a rough outline to be preferable showing the most important features of the time zone. More specifically, we compute all reachable and unreachable parts of the road network once at the beginning. When creating the time zone for a single zoom level, we then remove all roads that are not displayed for this particular zoom level and construct the time zone based on the remaining roads. Hence, the SEPARATION PROPERTY still holds for all drawn roads, but it may be violated for the removed roads. Further, the time zone directly adapts to the current selection of roads.

Figure 15 shows an example for a travel-time map for two zoom levels. In Figure 15a two areas with a large number of bends are highlighted. The first one shows a graveyard with small footpaths, while the second one shows a field path parallel to a residential road. In both cases, a detailed outline with a large number of bends is necessary to satisfy the SEPARATION PROPERTY. Figure 15b shows the generalized time zones of the same example. The visualization for both marked regions is simplified and the octilinear structure of the polygons is easily recognizable. In the given example, the blue area remains almost unchanged as no roads have been generalized here. In particular, the improvement of the SCHEMATIZATION relies on a good selection of roads to be removed.

7.3 | Clipping

We observe that time zones may overlay important natural barriers such as rivers and lakes; see Figure 4. However, this easily creates the impression that the region is easily accessible on foot. Furthermore, schematizing the outline of these natural barriers may compromise the "mental map" a user has of the area. We therefore clip the time zones by subtracting important natural barriers. Sometimes, this creates small scratches of the polygon on the wrong side of the river, in areas where no roads are present. As we do not have information about accessibility for regions without roads, we remove these shreds. With this, we obtain an appearance of the time map that blends in with the natural features of the map.

8 | EXPERIMENTS

In this section we describe our experiments on real-world data. In Section 8.1 we present the experimental setup. In Section 8.2 we discuss results based on a single travel time, comparing them to other visualization approaches for

travel-time maps. Finally, in Section 8.3, we present a small case study featuring a travel-time map with multiple time zones.

8.1 | Experimental setup

We have created travel-time maps for the metropolitan area of Cologne and Bonn, Germany. The area has a densely developed public transportation network that consists of local, regional and long-distance transport. Moreover, the Rhine, which is a major waterway in central Europe, passes through the region and separates it into an eastern and western part. Both parts are connected by 11 bridges and six ferries spread out over a length of 60 km along the river. Thus, the river constitutes a major barrier for accessibility.

Our input data stem from two sources: for the road network we use OSM data[1] extracted by Geofabrik (www.geofabrik.de). The extent of the data used is approximately 40 km from north to south and 60 km from east to west, centered on the metropolitan area of Cologne and Bonn. The public transportation network data are taken from OpenData VRS (www.vrs.de) in the GTFS format and contain 2018–2019 timetable information for local transportation as well as regional trains. Further, for our experiments we inserted data about ferries crossing the river Rhine.

For our evaluation we applied the algorithm for 20 different starting locations dispersed around the experimental region. We used Monday 9 a.m. as starting time and computed travel-time maps for eight travel times (in minutes), namely $T = \{5, 10, 15, 20, 25, 30, 35, 40\}$.

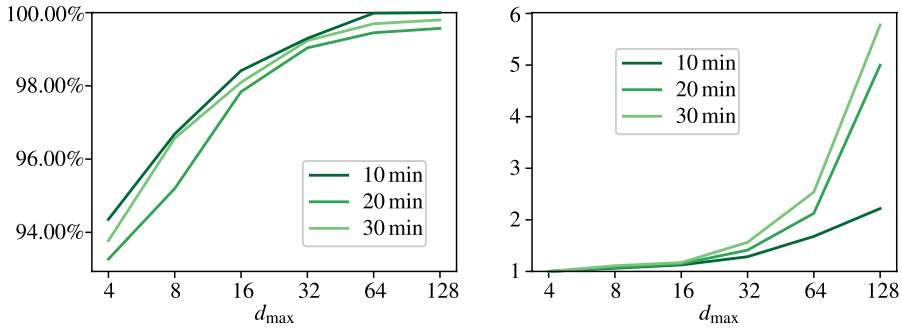
The implementation was written in Java and the experiments were performed on an AMD EPYC™ 7402P processor clocked at 2.8 GHz with 128 GB RAM. The planarization of the graph took 62 seconds and creating the remaining travel-time map took 18 seconds on average.

In order to decide on a suitable value for the maximum degree $d_{\max}$ of refinement we conducted initial experiments. To this end, we calculated time zones for three different travel times with $d_{\max}$ varying from $d_{\max} = 4$ to $d_{\max} = 128$; see Figure 16.

The higher the maximum degree of refinement is, the finer the octilinear grid gets. Hence, the share of the octilinear edges of the resulting polygon increases with increasing maximum degree of refinement. For $d_{\max} = 32$ over 99% of all edges are octilinear; see Figure 16a. Moreover, for $d_{\max} = 32$ the running time is increased by a factor of at most 1.56 compared to $d_{\max} = 4$; see Figure 16b. Thus, for the experiments we have chosen $d_{\max} = 32$ as a suitable compromise between accuracy and running time.

8.2 | Evaluation

We evaluate the results of our approach by comparing them with the results of other visualization techniques for travel-time maps. Specifically, the techniques we compare our approach to are the *time-buffered visualization* inspired by our initial example for the city of Melbourne (see Figure 2); the *arboreal visualization* as presented by Marciuska and Gamper (2010) using a very small buffer size to best ensure the SEPARATION PROPERTY; and the *minimum-perimeter visualization* which is a slight adaptation of the technique introduced by Baum et al. (2018), minimizing the perimeter instead of the number of bends. Note, however, that we are using our own implementations of the respective works, and results can differ from results generated by the original solution. We discuss the results with respect to their basic properties as well as the visual complexity. As simple measures for the visual complexity of a polygon we use its perimeter, area, the type of angles (acute or obtuse) and the directions of its edges. These measures are a common tool to assess the complexity of geometric shapes (Chen & Sundaram, 2005).



(a) octilinearity (b) running Time

FIGURE 16 Evaluation of the maximum degree of refinement d_{max}. (a) Share of the boundary's length that is visualized octilinearly. (b) Running time compared to the running time using the lowest d_{max}

8.2.1 | Octilinear visualization versus time-buffered visualization

First, we compare the octilinear visualizations with time-buffered visualizations. At the core of this visualization technique is a constant off-road speed, which is assumed to be lower than the walking speed on roads. For each point of the road network (no matter whether it is a node or a point on an edge) the remaining travel time at this point is visualized by a buffer computed by means of the off-road speed; see Figure 17b. Consequently, the SEPARATION PROPERTY is not enforced by this visualization technique. While reachable parts of the road network are guaranteed to be contained in the time zone, unreachable parts may also be covered by the zone, violating our SEPARATION PROPERTY requirement, as seen in Figure 3a. Another example is shown in Figure 17b ① and ② where parts of the road network that are not reachable are covered by the zone. Similar examples are easily found all along the zone's boundary. For the given example, in total 72 nodes of the road network are wrongly represented as reachable. In contrast, the octilinear time zone shown in Figure 17a precisely separates the roads, as required in our algorithm. Moreover, the time-buffered visualization tends to become frayed at the boundary as highlighted in Figure 17b ②, also violating our LOW VISUAL COMPLEXITY requirement.

We insist on the SEPARATION PROPERTY as the main requirement for the travel-time maps. In the following we therefore only evaluate approaches that fulfill this requirement.

8.2.2 | Octilinear visualization versus arboreal visualization

The second approach visualizes the outer boundary of the reachable zone, which we call *arboreal visualization* due to the ramified shape of the time zones; see Figure 17c. We emphasize that this representation gets cluttered along the boundaries of the zone, since the last reachable roads form tree-like dangles attached to a polygon. Hence, in contrast to our approach, this kind of visualization tends to substantially larger perimeters; see Figure 18a. We observe that especially for larger travel times the perimeter differs from our approach; for example, for a travel time of 30 min the perimeter is increased by a factor of 2.9.

To analyze the types of the angles and the direction of the time zones' edges we create a histogram of the outer angles of the polygons; see Figure 19. To that end, we split the angle interval $[0, 2\pi]$ into 16 equal-sized bins such that the ith bin is defined by the interval $I_i = [(2i-3)\pi/16, (2i-1)\pi/16)$, for $1 \leq i \leq 16$. With this definition the octilinear directions do not form the boundaries of these intervals, but split them into equal-sized subintervals. For each bin i we count how many bends of the polygon have an outer angle that lies in I_i.

We observe that the number of acute angles (three backward-facing bins, outer angle smaller than $33.75°$ or greater than $326.25°$) is significantly higher for the arboreal visualization than for the octilinear visualizations;

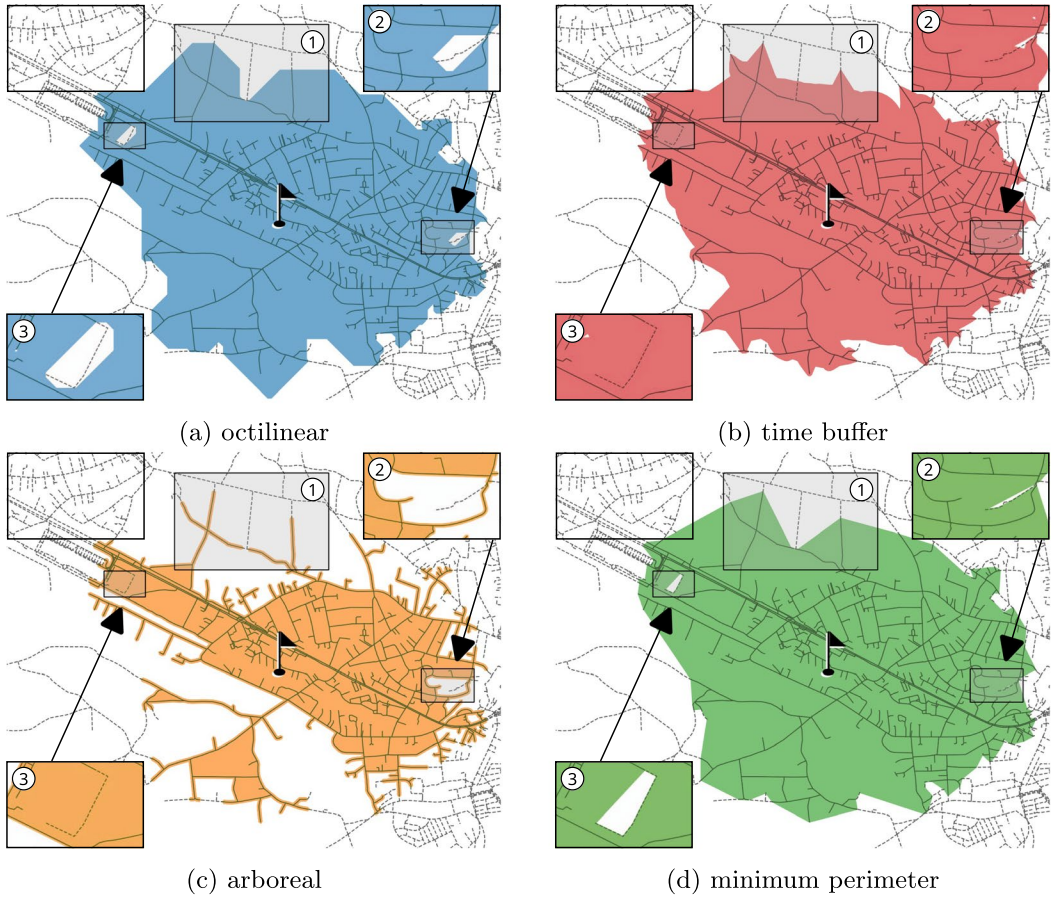


FIGURE 17 Single time zone for $\tau = 15$ minutes for different visualization types. and show areas where the SEPARATION PROPERTY is violated in (a) and (b); highlights a hole which is hard to recognize using some visualizations such as (c)

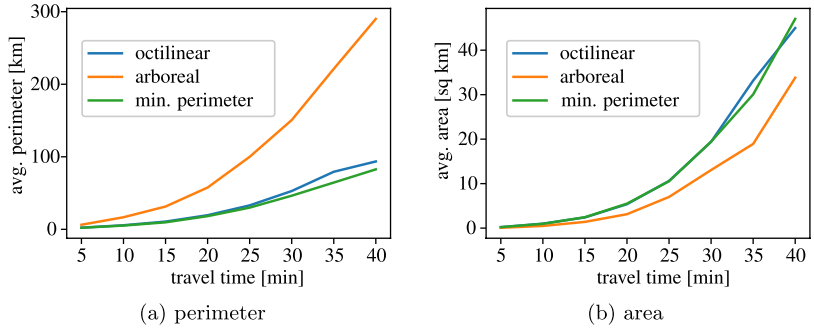
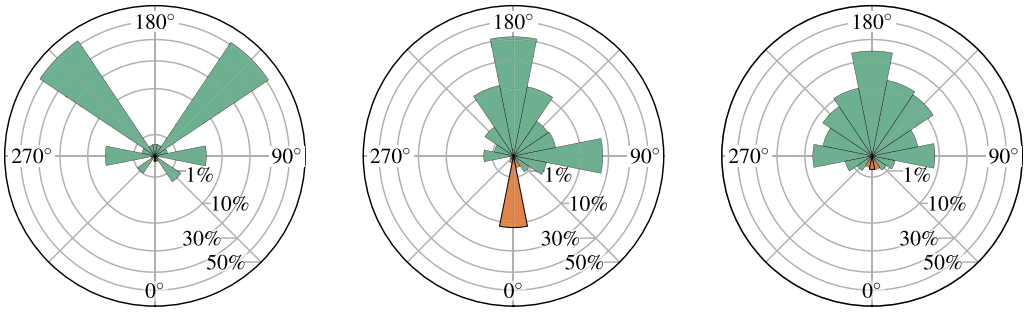


FIGURE 18 (a) Perimeter and (b) area for different visualization types

see Figure 19b. The share of acute angles int the total number of angles is 12.2 %, while for our approach it is only 0.07 %. This supports the visual impression of Figure 17 that the complexity for arboreal visualizations is much higher than for octilinear visualizations.



(a) octilinear (b) arboreal (c) minimum perimeter

FIGURE 19 Histogram of the time zone's outer angles, averaged over all starting points for a travel time of $\tau = 30$ min. Orange bins relate to acute angles. The number per bin is proportional to the bin's area

8.2.3 | Octilinear visualization versus minimum-perimeter visualization

Finally, we compare our approach with visualizations that minimize the perimeter of the travel-time zones; see Figure 17d. The approach that is presented by Baum et al. (2018) yields visually similar results. We first observe that the minimum-perimeter and octilinear visualizations lead to similar perimeters; see Figure 18a. For our approach the perimeters are on average 11 % longer and at most 23 % longer. Moreover, the area covered by the time zones is comparable in each case, which indicates relatively small differences between the time zones. However, evaluating the distribution of the outer angles, the effect of the SCHEMATIZATION in our approach becomes evident; see Figure 19. Over 98.5 % of all outer angles lie in bins corresponding to the octilinear directions. In particular, the vast majority (83.7 %) lie in the bin that represents the two obtuse angles 225° and 135°. In contrast, for the minimum-perimeter visualization the angles are spread across the whole histogram, leading to a higher visual complexity with regard to this criterion.

Similarly to the arboreal visualization, the number of acute angles is higher for the minimum-perimeter visualization than for the octilinear visualization. The share of acute angles is over nine times higher than for our approach. These acute angles can create artifacts in the visualization, where parts of the polygon are hard to spot. In Figure 17d the time zone has a hole at , but as it is visualized with a thin polygon having acute angles at its bends, it can hardly be spotted. In contrast, the hole is clearly visible in the octilinear time zone; see Figure 17a.

The analysis shows that the octilinear visualization has a lower visual complexity than the minimum-perimeter visualization. This coincides with the visual impression of the maps shown in Figure 17. Nevertheless, there may be cases where the octilinear visualization suffers from stair-shaped lines in graph faces with unfavorable geometry. An example is given in Figure 20. These unfavorable geometries occur for long stretched faces whose main direction is not one of the octilinear directions. In contrast, the minimum-perimeter visualization has one straight edge at this location. A combination of the two visualization approaches thus could lead to an even better result. For example, we could check if the number of bends in an octilinear face exceeds a threshold k and then fall back to the minimum-perimeter visualization for this face. An investigation of the combination of both algorithms is left to future work.

8.3 | Case study

We conclude the evaluation with a small case study highlighting the important steps for creating travel-time map with multiple time zones; see Figures 21 and 22. To that end, we used a starting location in Wesseling, situated between Cologne and Bonn, to obtain four time zones. Creating each time zone separately might cause

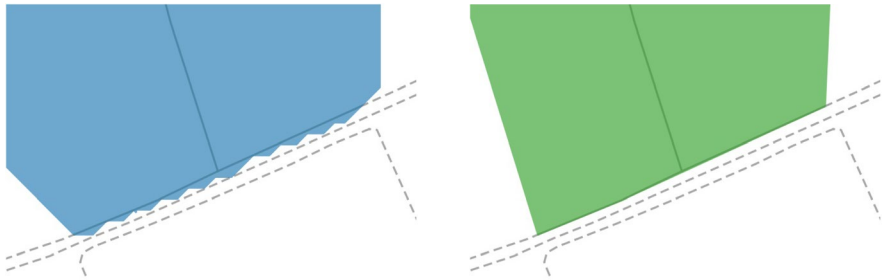


FIGURE 20 In rare cases (a) the octilinear visualization has a step-like outline, while (b)the minimum-perimeter visualization does not suffer from this

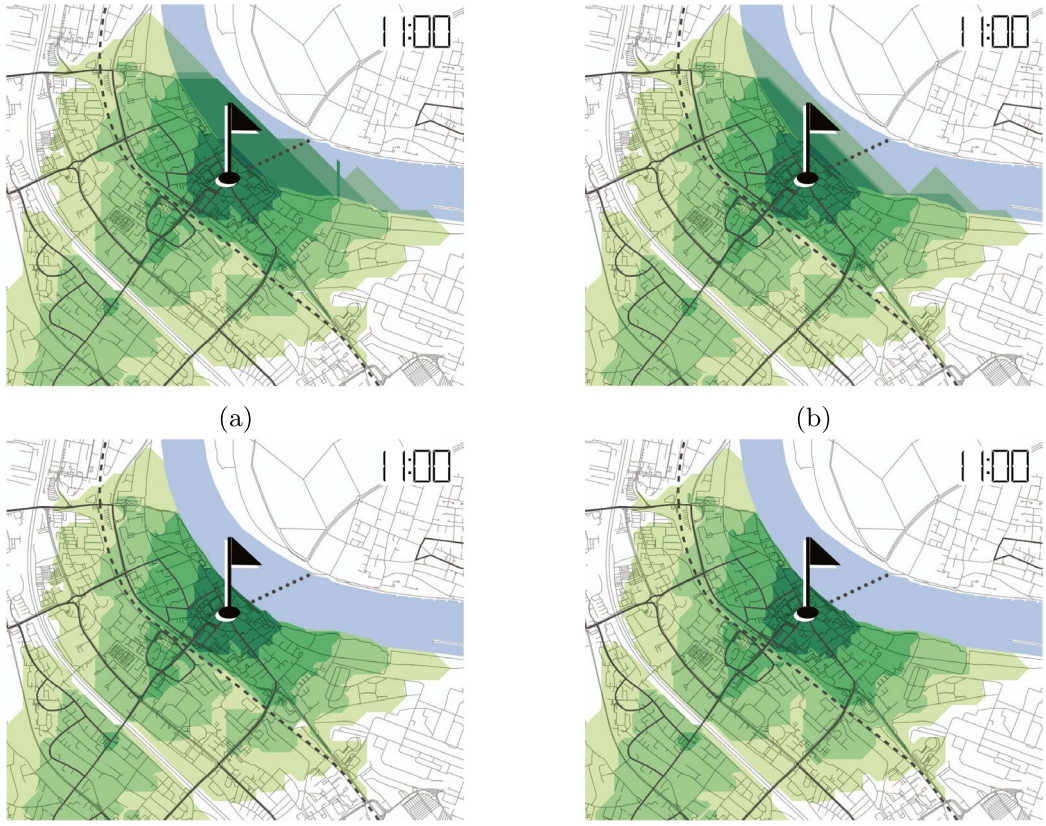


FIGURE 21 Case study. The travel-time map is created for four time zones with $T = \{5, 10, 15, 20\}$ minutes and start time Monday 11 a.m. (a) Independently generated time zones. (b) Time zones nested in the next bigger time zone. (c) Clipping to the river. (d) Result after applying the closing operation

inner time zones overlapping the outer ones if their boundaries share the same graph faces, see Figure 21a. To eliminate this issue, the outer time zones are used as a spatial limit for the inner ones. With this the inner zones strictly lie inside the outer ones, which leads to a less cluttered map; see Figure 21b. Nonetheless, it remains non-intuitive that time zones are differentiated on the Rhine. To remove this issue we deploy our clipping extension (see Section 7.3) in which we remove all parts of the time zones covering such natural

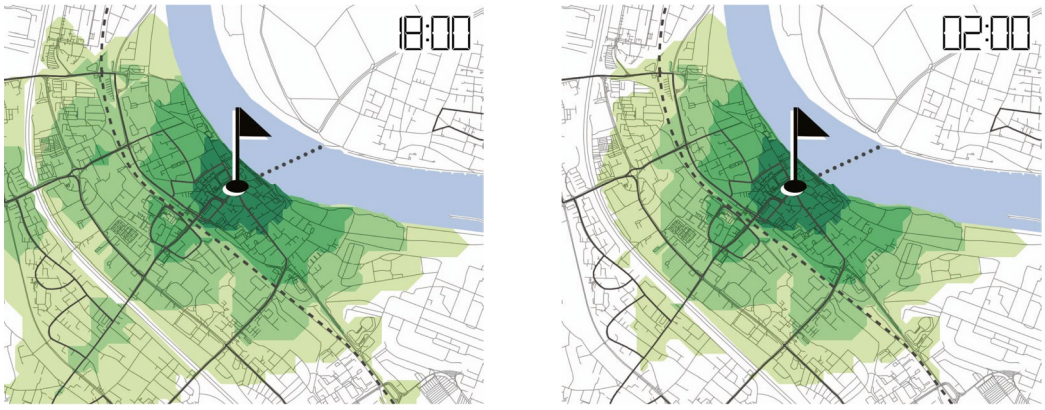


FIGURE 22 Continuation of the case study. (a) Travel times with start time Monday 6 p.m. (b) Travel times with start time Tuesday 2 a.m.

boundaries. As a result, the time zones represent the topography better and thus become more comprehensible; see Figure 21c.

As a last processing step, we apply the closing extension (see Section 7.1) to eliminate small holes in the time zones; see Figure 21d. We have used a dilation factor of 10 %, that is, a hole is closed if it is completely covered by a travel time increased by 10 % of the original travel time. This further reduces the visual complexity of the resulting travel-time map. In the context of public transportation, this closing operation can be interpreted as a modeled vagueness in the travel-time map due to delays or varying walking speeds.

Thanks to the integration of timetable information of the public transportation, the travel-time map can not only be generated for different starting points and travel times, but also for different starting times. Examples of the variation in travel-time zones depending on the time of the day are shown in Figure 22.

9 | CONCLUSIONS

This article presented an automatic approach for the generation of schematic travel-time maps. From a user location, outlines of reachable regions are generated for a certain travel time. Following the spirit of schematized metro maps, our method generates octilinear polygonal zones. This reduces the visual complexity and leads to a higher map legibility attributed to an optimization of the number of polygon bends.

We conclude that the computation of schematic travel-time maps can be modeled as a search for a minimum-weight path in an appropriately defined graph. In particular, using the *line graph* of a graph obtained from augmenting a regular octilinear grid with additional segments through ports (i.e., points where the unreachable and reachable part of a network meet) allowed us to heuristically minimize the number of bends of the isochrones.

We further showed how to apply this approach to generate travel-time maps for multiple travel times and showed its feasibility through a case study. The evaluation of the results has shown that the different visualization approaches for travel-time maps have their own advantages and disadvantages. Future research should involve the combination of these approaches to benefit from these respective advantages. For example, one could combine our approach with the approach by Baum et al. (2018) to reduce sawtooth patterns in the visualization. Moreover, we deem the combination of our approach with schematized networks maps as fruitful. For example, one may use our approach to create schematized fare zones in schematized metro maps.

ORCID

Axel Forsch http://orcid.org/0000-0002-3849-4865

ENDNOTE

[1] ©OpenStreetMap contributors.

REFERENCES

Bauer, V., Gamper, J., Loperfido, R., Profanter, S., Putzer, S., & Timko, I. (2008). Computing isochrones in multi-modal, schedule-based transport networks. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Irvine, CA (pp. 1–2). New York, NY: ACM. https://doi.org/10.1145/1463434.1463524.

Baum, M., Bläsius, T., Gemsa, A., Rutter, I., & Wegner, F. (2018). Scalable exact visualization of isocontours in road networks via minimum-link paths. *Journal of Computational Geometry*, *9*, 24–70. https://doi.org/10.20382/jocg.v9i1a2

Baum, M., Buchhold, V., Dibbelt, J., & Wagner, D. (2019). Fast exact computation of isocontours in road networks. *ACM Journal of Experimental Algorithmics*, *24*, 1–18. https://doi.org/10.1145/3355514

Bonerath, A., Niedermann, B., & Haunert, J. H. (2019). Retrieving alpha-shapes and schematic polygonal approximations for sets of points within queried temporal ranges. *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, IL (pp. 249–258). New York, NY: ACM. https://doi.org/10.1145/3347146.3359087.

Bouts, Q. W., Kostitsyna, I., van Kreveld, M., Meulemans, W., Sonke, W., & Verbeek, K. (2016). Mapping polygons to the grid with small Hausdorff and Fréchet distance. In P. Sankowski, & C. Zaroliagis (Eds.), *Proceedings of the 24th Annual European Symposium on Algorithms* (pp. 22:1–22:16). Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ESA.2016.22

Brewer, C. A., Stanislawski, L. V., Buttenfield, B. P., Sparks, K. A., McGilloway, J., & Howard, M. A. (2013). Automated thinning of road networks and road labels for multiscale design of the national map of the United States. *Cartography and Geographic Information Science*, *40*, 259–270. https://doi.org/10.1080/15230406.2013.799735

Buchin, K., Meulemans, W., Renssen, A. V., & Speckmann, B. (2016). Area-preserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, *2*(1), 2. https://doi.org/10.1145/2818373

Chen, Y., & Sundaram, H. (2005). Estimating complexity of 2D shapes. In *Proceedings of the Seventh IEEE Workshop on Multimedia Signal Processing*, Shanghai, China (pp. 1–4). Piscataway, NJ: IEEE.

Chimani, M., van Dijk, T. C., & Haunert, J. H. (2014). How to eat a graph: Computing selection sequences for the continuous generalization of road networks. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Dallas, TX (pp. 243–252). New York, NY: ACM. https://doi.org/10.1145/2666310.2666414

Gamper, J., Böhlen, M., Cometti, W., & Innerebner, M. (2011). Defining isochrones in multimodal spatial networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, Glasgow, UK (pp. 2381–2384). New York, NY: ACM. https://doi.org/10.1145/2063576.2063972

Gamper, J., Böhlen, M. H., & Innerebner, M. (2012). Scalable computation of isochrones with network expiration. In A. Ailamaki, & S. Bowers (Eds.), *Scientific and statistical database management: SSDBM 2012* (Lecture Notes in Computer Science, Vol. *7338*, pp. 526–543). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-31235-9_35.

Haunert, J. H., & Wolff, A. (2010). Optimal and topologically safe simplification of building footprints. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Jose, CA (pp. 192–201). New York, NY: ACM. https://doi.org/10.1145/1869790.1869819

Huang, W., Hong, S. H., & Eades, P. (2008). Effects of crossing angles. In *Proceedings of the 2008 IEEE Pacific Visualization Symposium*, Kyoto, Japan (pp. 41–46). Piscataway, NJ: IEEE. https://doi.org/10.1109/PACIFICVIS.2008.4475457

Imai, H., & Iri, M. (1988). Polygonal approximations of a curve–formulations and algorithms. *Machine Intelligence and Pattern Recognition*, *6*, 71–86. https://doi.org/10.1016/B978-0-444-70467-2.50011-4

Krajzewicz, D., & Heinrichs, D. (2016). UrMo accessibility computer—A tool for computing contour accessibility measures. In *Proceedings of the Eighth International Conference on Advances in System Simulation*, Rome, Italy (pp. 56–60).

Krismer, N., Silbernagl, D., Specht, G., & Gamper, J. (2017). Computing isochrones in multimodal spatial networks using tile regions. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, Chicago, IL (pp. 1–6). New York, NY: ACM. https://doi.org/10.1145/3085504.3085538

Krismer, N., Specht, G., & Gamper, J. (2014). Incremental calculation of isochrones regarding duration. In *Proceedings of the 26th GIWorkshop on Foundations of Databases*, Bozen, Italy (pp. 41–45).

Marciuska, S., & Gamper, J. (2010). Determining objects within isochrones in spatial network databases. In B. Catania, M. Ivanović, & B. Thalheim (Eds.),*Advances in databases and information systems: ADBIS 2010* (Lecture Notes in Computer Science, Vol. *6295*, pp. 392–405). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-15576-5_30

Nöllenburg, M. (2014). A survey on automated metro map layout methods. In *Proceedings of the 2014 Schematic Mapping Workshop*, Essex, UK (pp. 1–7).

O'Sullivan, D., Morrison, A., & Shearer, J. (2000). Using desktop GIS for the investigation of accessibility by public transport: An isochrone approach. *International Journal of Geographical Information Science*, *14*, 85–104. https://doi.org/10.1080/136588100240976

Purchase, H. C. (2000). Effective information visualisation: A study of graph drawing aesthetics and algorithms. *Interacting with Computers*, *13*, 147–162. https://doi.org/10.1016/S0953-5438(00)00032-1

Pyrga, E., Schulz, F., Wagner, D., & Zaroliagis, C. (2008). Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, *12*. https://doi.org/10.1145/1227161.1227166

Serra, J. (1983). *Image analysis and mathematical morphology*. San Diego, CA: Academic Press.

Ware, C., Purchase, H. C., Colpoys, L., & McGill, M. (2002). Cognitive measurements of graph aesthetics. *Information Visualization*, *1*, 103–110. https://doi.org/10.1057/palgrave.ivs.9500013

Wu, H. Y., Niedermann, B., Takahashi, S., & Nöllenburg, M. (2019). A survey on computing schematic network maps: The challenge to interactivity. In *Proceedings of the Second Schematic Mapping Workshop*, Vienna, Austria (pp. 1–7).

Wu, H. Y., Niedermann, B., Takahashi, S., Roberts, M. J., & Nöllenburg, M. (2020). A survey on transit map layout - from design, machine, and human perspectives. *Computer Graphics Forum*, *39*, 619–646. https://doi.org/10.1111/cgf.14030

Zhou, Q., & Li, Z. (2016). Empirical determination of geometric parameters for selective omission in a road network. *International Journal of Geographical Information Science*, *30*, 263–299. https://doi.org/10.1080/13658816.2015.1085538

How to cite this article: Forsch, A., Dehbi, Y., Niedermann, B., Oehrlein, J., Rottmann, P., & Haunert, J. H. (2021). Multimodal travel-time maps with formally correct and schematic isochrones. *Transactions in GIS*, 25, 3233–3256. https://doi.org/10.1111/tgis.12821