ISOGA: A System for Geographical Reachability Analysis[⋆]

Markus Innerebner[1], Michael Böhlen[2], and Johann Gamper[1]

[1] Free University of Bozen-Bolzano, Italy
[2] University of Zurich, Switzerland

Abstract. In this paper, we present a web-based system, termed ISOGA, that uses isochrones to perform geographical reachability analysis. An isochrone in a spatial network covers all space points from where a query point is reachable within given time constraints. The core of the system builds an efficient algorithm for the computation of isochrones in multimodal spatial networks. By joining isochrones with other databases, various kinds of geospatial reachability analysis can be performed, such as how well is a city covered by public services or where to look for an apartment at moderate prices that is close to the working place. ISOGA adopts a service-oriented three-tier architecture and uses technologies that are compliant with OGC standards. We describe several application scenarios in urban and extra-urban areas, which show the applicability of the tool.

Keywords: Spatial networks, isochrones, geospatial reachability analysis, WebGIS.

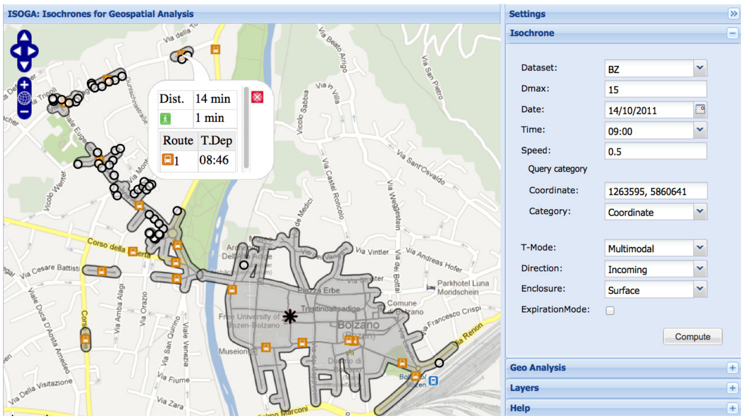
1 Introduction

Geospatial analysis covers various approaches to perform analysis on data with a geographical dimension and provides an important tool in many application areas, including environmental sciences, social sciences, emergency management, or city planning.

In this paper, we describe ISOGA, a system for geographical reachability analysis using isochrones in multimodal spatial networks. An *isochrone* in a spatial network is a possibly disconnected subgraph that covers all space points from where a query point q is reachable within a given time span and by a given arrival time at q. As an example, consider a person looking for an apartment in a specific price range, from where his/her working place is reachable in less than 15 minutes using the public transportation system. Figure 1 illustrates this query for Bozen-Bolzano. The '*' indicates the working place (query point), the gray area represents the isochrone, and white circles represent buildings that satisfy the search criteria. The popup shows additional information about one of the qualifying apartments, such as the actual distance and details on how (bus numbers and departure times) to reach the working place.

In ISOGA, the user inputs first the parameters for an isochrone, i.e., one or more query points, walking speed, arrival time, and a maximal timespan. At the core of the system is an efficient algorithm for the computation of isochrones in multimodal spatial

[⋆] This work is partially funded by the Province of Bozen-Bolzano as part of the AQUIST project.



ISOGA: Isochrones for Geospatial Analysis

Dist.	14 min
🚶	1 min
Route	T.Dep
🚌1	08:46

Settings

Isochrone

Dataset: BZ

Dmax: 15

Date: 14/10/2011

Time: 09:00

Speed: 0.5

Query category

Coordinate: 1263595, 5860641

Category: Coordinate

T-Mode: Multimodal

Direction: Incoming

Enclosure: Surface

ExpirationMode: ☐

Fig. 1. A 15 minute Isochrone in Bolzano-Bozen

networks together with the possibility to join an isochrone with an arbitrary relation of geo-referenced objects, e.g., people, houses, or hotels. First, the disk-based algorithm MINEX computes an isochrone as a subgraph of the multimodal network. Second, the surface of the isochrone subgraph is determined. Third, the isochrone surface is joined with a relation of geo-referenced objects, which can be specified by the user as an arbitrary SQL expression. As the result of a query, a simple summary statistics is shown together with a list of all objects that are located within the isochrone. The objects can be visualized on the interactive map as well, and by clicking on an object a popup shows additional information. ISOGA adopts a service-oriented three-tier architecture and uses standardized OGC services for exchanging spatial data. The system can be accessed at `www.isochrones.inf.unibz.it/isoga`. To summarize, the main contributions of this paper are as follows:

- We present the ISOGA system for geographical reachability analysis in multimodal networks, which uses isochrones to efficiently determine geo-referenced objects that are reachable within given time constraints.
- We describe the three-tier architecture of the system with an interactive WebGIS client that uses OGC standards for the communication between client and server.
- We discuss three application scenarios using real-world data that illustrate the applicability of ISOGA for various kinds of geographical reachability analysis.

The rest of the paper is structured as follows. Section 2 discusses related work. In Section 3 we describe the scientific background of the system. The system architecture is presented in Section 4. Section 5 describes three application scenarios.

2 Related Work

Previous work on isochrones, upon which this paper is based, has been presented in [1, 4, 5, 7]. Bauer et al. [1] introduce a main memory algorithm that suffers from a high initial loading cost and is limited by the available memory. To overcome these limitations, Gamper et al. [4] propose a disk-based algorithm that loads the network incrementally and is independent of the network size. This work is extended in [5] with network expiration, which allows to keep in memory only the expansion frontier to avoid cyclic expansions. Marciuska and Gamper [7] present two approaches to transform an isochrone subgraph into a spatial area that is needed for joins with other spatial objects. The architecture of the ISOGA system follows the WebGIS framework in [6].

Different query types have been studied for spatial network databases, e.g., [2,3,8]. Isochrones are closest to range queries which return all objects that are within a given distance, whereas an isochrone query returns all *space points* within a given distance. Isochrone queries are more flexible than range queries. Once an isochrone is computed, it can be reused to retrieve any kind of geo-referenced objects that are located within the isochrone without the need to compute the actual distance to these objects.

The project pgRouting[1] extends the spatial DBMS PostGIS with geospatial routing functionalities. The driving distance (isoline) function supports range queries with a dynamic cost parameter, but it does not support schedule-based networks. Mapnificient[2] uses a simple heuristic based on the Euclidean distance to approximate isochrones. Network expansion examines only the transportation network, whereas the reachable areas in the pedestrian network are approximated by drawing a circle around bus stops with a radius that is determined by the available time. A similar approach is adopted in Mapumental[3], which uses isochrones for house hunter services. The company Hacon[4] offers the product Hafas that computes range queries in transportation networks. The trip planner OpenTripPlanner[5] provides an extension for isochrones to measure the accessibility to or from specific locations as well as to perform aggregate search analysis.

The ISOGA system described in this paper integrates previous work on isochrones into a tool for geographical reachability analysis. Different from other systems, ISOGA works for multimodal networks that represent different transportation modes, and it efficiently computes exact isochrones rather than approximate solutions. The geo-referenced objects used in the analysis can be specified by an arbitrary SQL query.

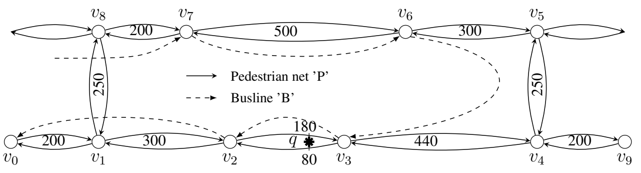
3 Scientific Background

3.1 Isochrones in Multimodal Networks

A *multimodal network* is defined as a seven-tuple $\mathsf{N} = (G, R, S, \rho, \mu, \lambda, \tau)$. G is a directed multigraph with a set V of vertices and a multiset E of edges. Vertices represent crossroads of the street network, stops of the public transport system, etc. Edges represent street segments, transport routes, moving walkways, etc. R is a set of transport systems, such as the pedestrian network or the public transport system (buses, trains, etc.). Function μ assigns to each transport system a transport mode, e.g., continuous space and time mode ('*csct*') for the pedestrian network or discrete space and time mode ('*dsdt*') for the public transport system. The functions ρ and λ assign to each edge transport system and edge length, respectively. Finally, function $\tau(e, t)$ computes the time-dependent transfer time that is required to traverse edge $e = (u, v)$ when starting at u as late as possible yet arriving at v no later than time t.

[1] `www.pgrouting.org`
[2] `www.mapnificent.net`
[3] `http://mapumental.com/`
[4] `www.hacon.de`
[5] `www.opentripplanner.org`

Figure 2 shows a multimodal network with two transportation systems, $R = \{\text{'P'}, \text{'B'}\}$, representing the pedestrian network with mode $\mu(\text{'P'}) = \text{'}csct\text{'}$ and bus line B with mode $\mu(\text{'B'}) = \text{'}dsdt\text{'}$, respectively. Solid lines are street segments of the pedestrian network, e.g., edge $e = (v_1, v_2)$ with $\rho(e) = \text{'P'}$. Pedestrian edges are annotated with the edge length, which is the same in both directions, e.g., $\lambda((v_1, v_2)) = \lambda((v_2, v_1)) = 300$. We assume a constant walking speed of 2 m/s, yielding a transfer time $\tau(e, t) = \frac{\lambda(e)}{2\,\text{m/s}}$ for a pedestrian edge e. Dashed lines represent bus line B. An excerpt of the schedule is shown in Figure 2(b). The transfer time of a bus edge $e = (u, v)$ is computed as $\tau(e, t) = t - t'$, where t' is the latest departure time at u in order to reach v before or at time t. This might include a waiting time at v.



(a) Network

TID	Stop	Arrival	Departure
1	v_7	05:31:30	05:32:00
1	v_6	05:33:00	05:33:00
1	v_6	05:35:00	05:35:00
⋮	⋮	⋮	⋮

(b) Schedule

Fig. 2. Multimodal Network

A *location* in N is any point on an edge $e = (u, v) \in E$ that is accessible. We represent it as $l = (e, o)$, where $0 \leq o \leq \lambda(e)$ is an offset that determines the relative position of l from u on e, e.g., the location of q in Figure 2 is $l_q = ((v_2, v_3), 180) = ((v_3, v_2), 80)$. In continuous space networks all points on the edges are accessible. In discrete space networks only vertices are accessible.

The *network distance*, $d_N(l_s, l_d, t)$, from a source location l_s to a destination location l_d with arrival time t at l_d is defined as the minimum cost of any path from l_s to l_d with arrival time t at l_d if such a path exists, and ∞ otherwise. The network distance is time-dependent. For instance, $d_N(v_6, v_3, \text{05:33:00}) = 120\,\text{s}$ because the bus with trip id 1 departs from v_6 at 05:33:00 and arrives at v_3 at 05:35:00. In contrast, $d_N(v_6, v_3, \text{05:34:00}) = 495\,\text{s}$ since the shortest path passes through the pedestrian network, traversing the edges (v_6, v_5), (v_5, v_4), and (v_4, v_3).

An *isochrone*, $N^{iso} = (V^{iso}, E^{iso})$, is defined as the minimal and possibly disconnected subgraph of G that covers exactly those locations that have a network distance to q smaller or equal than a user-defined timespan d_{max}. In Figure 3, the boldface edges (edge segments) represent an isochrone, which is formally represented by the vertices $V^{iso} = \{v_3, v_2, v_3, v_6, v_1, v_7, v_4\}$ and the edges (edge segments) $E^{iso} = \{((v_0, v_1), 80, 200), ((v_8, v_1), 130, 250), ((v_2, v_1), 180, 300), ((v_1, v_2), 0, 300), ((v_3, v_2), 0, 260), \ldots\}$.

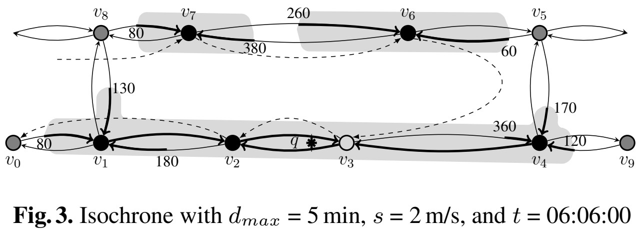


Fig. 3. Isochrone with d_{max} = 5 min, s = 2 m/s, and t = 06:06:00

For the computation of isochrones, we use the MINEX algorithm [5], which adopts an incremental network expansion strategy in combination with vertex expiration. The expansion starts from q and propagates backwards along the incoming edges in all directions. The multimodal network is stored in the database, and – as the expansion proceeds – the portions of the network that eventually will form the isochrone are incrementally retrieved. By loading only those network portions that eventually will be part of the isochrone and by keeping in memory only a minimal set of vertices to avoid cyclic network expansion, MINEX is independent of the network size and the memory requirements are only a small fraction of the isochrone size.

3.2 Creating the Surface of an Isochrone Subgraph

In order to determine objects that are located within isochrones, we have to construct an area around an isochrone. Marciuska and Gamper [7] propose two algorithms for this. The surface-based approach (SBA) computes a minimum bounding polygon around the outermost edges in the isochrone. However, it is limited to isochrones in pedestrian networks, where an isochrone consists of a single subgraph.

Algorithm SBA^* in Figure 4 extends SBA by allowing isochrones that are composed of disconnected subgraphs, which is typically the case if public transport systems are present in addition to the pedestrian network. The input parameters are a set of edges E^{iso} that represents an isochrone and a parameter $size$ that represents the margin of the buffer that is created around the outermost edges. SBA^* iterates over set E^{iso} and computes in each iteration an area around a single subgraph of the isochrone. The algorithm identifies first the leftmost edge e in E^{iso} and calls then function DFS, which in a recursive way traverses the outermost edges of the subgraph that contains e. Parameter P returns the ordered list of points that represent the geometry of the

```
**Algorithm:** $SBA^*(E^{iso}, size)$
while $E^{iso} \neq \emptyset$ do
    $P \leftarrow \emptyset$;
    $e = (u, v) \leftarrow$ leftmost edge in $E^{iso}$;
    $DFS(e, e, E^{iso}, P, 0)$;
    $g \leftarrow ST\_Buffer(ST\_MakePolygon(P), size)$;
    $E^{iso} \leftarrow E^{iso} \setminus ST\_Within(E^{iso}.geo, g)$;
    Output $ST\_Multi(g)$;

**Function:** $DFS(root, e, E^{iso}, P, l)$
if $e = root \wedge l > 0$ then
    $P \leftarrow P \cup geom(e)^{-1}$;
    return $true$;
else
    $(u, v) \leftarrow e$;
    if $l = 0$ then
        $P \leftarrow P \cup geom(e)$;
        foreach $(u', v) \in E^{iso}$ sorted by angle $\alpha((u, v), (u', v))$ do
            if $DFS(root, (u, v), E^{iso}, P, l + 1)$ then return $true$;
            else $P \leftarrow P \cup geom(e)$;
        return $false$;
    else
        if $o(e) + o(e^{-1}) \leq \lambda(e)$ then
            $P \leftarrow P \cup geom(e)^{-1} \cup geom(e^{-1}))$;
            foreach $(t, u) \in E^{iso}$ sorted by angle $\alpha((u, v), (t, u))$ do
                if $DFS(root, (t, u), E^{iso}, P, l + 1)$ then return $true$;
                else $P \leftarrow P \cup geom(e)$;
        else
            $P \leftarrow P \cup geom(e)^{-1}$;
        return $false$;
```

Fig. 4. Algorithm SBA^*

outermost edges. P is transformed into a polygon, and a buffer with a margin of size $size$ is created around the polygon. Finally, all edges that are within the isochrone area are removed from E^{iso}.

Function DFS recursively computes the outermost edges of an isolated subgraph and collects the geometry of these edges in an ordered set of points P. DFS has in input the leftmost edge $root$, the currently visited edge e, the isochrone edges E^{iso}, the ordered set of points P, and the recursion level l. DFS examines in a depth-first traversal the subgraph until it returns to the root edge (i.e., the left-most edge) or no other edges are found. If $e = root$ and $l > 0$, the traversal returned to the root edge. The geometry of the current edge is added to P (after converting the order of the points) and the recursion terminates. Otherwise, if the recursion level l is equal to 0, the geometry of the current edge $e = (u, v)$ is added to P. Then, all incoming edges to v, ordered by the angle to e, are considered by recursively calling DFS with a recursion level that is incremented by one. If DFS returns true, the recursion is stopped. Otherwise, the geometry of e is added to P, and the next incoming edge is considered. If $l > 0$, the edges are processed in a similar way. The only difference is that we have to consider the case that an edge might only be partially reachable; $o(e)$ is the offset of the reachable segment from the source vertex of e and $o(e^{-1})$ is the offset of the reachable edge segment from the target vertex of e. If $o(e) + o(e^{-1}) \leq \lambda(e)$, the two segments cover the entire edge, hence the recursive traversal of the edges continues. If this is not the case, only the segment of the current edge is added and the recursion stops.

We illustrate algorithm SBA^* using the isochrone in Figure 3. The first leftmost edge that is identified is (v_0, v_1). It is passed to DFS together with E^{iso}, an empty set P, and recursion level 0. DFS adds the geometry of the edge to P and iterates then through all incoming edges to v_1, sorted by their relative (counterclockwise) angle with respect to (v_0, v_1), i.e., (v_2, v_1), $(v_8, v1)$, and $(v_0, v1)$ in that order. Next, DFS is called with $e = (v_2, v_1)$, the current set P, and recursion level 1. The recursive traversal continues until edge (v_9, v_4) is encountered, which is only partially reachable. DFS returns by one recursion level and examines edge (v_5, v_4), etc. Finally, the root edge is visited again, and DFS returns to SBA^*. The points in P are transformed into a polygon, around which a buffer with a margin of size $size$ is created (light-gray area in Figure 3). After removing all edges from E^{iso} that are enclosed in the new isochrone area, the next leftmost edge (v_8, v_7) is determined, followed by a call to DFS, etc.

4 Architecture

Figure 5 shows the three-tier architecture of the ISOGA system.

Presentation Tier. The presentation tier is a WebGIS client, implemented in JSP and JavaScript. It uses *Comet*[6] as web application model for asynchronous data sending and for managing long polling requests, *Openlayers*[7] as web mapping framework, and *GeoExt*[8] as framework for building interactive web applications. The main tasks of the client is the interaction with the map, the input of the query parameters, and the

[6] www.cometd.org
[7] www.openlayers.org
[8] www.geoext.org

visualization of the results. The client communicates with the server over the HTTP protocol using the following standardized OGC[9] services: Web Map Service (*WMS*) for serving geo-referenced map images and Web Feature Service (*WFS*) for requesting geographical features. The client submits asynchronously three different types of HTTP requests. An *isochrone request* ① invokes MINEX for the computation of an isochrone. A *map request* ② retrieves the isochrone in form of a binary image format and includes it as a separate layer in the map. Similar, the base layer images (e.g., the street network) come from different sources (Google, OpenStreetMap) and are fetched via WMS. The request is triggered during the initialization of the map, when MINEX and SBA^* are terminated, or whenever there is an interaction with the map (zoom, pan, identify). A *feature request* ③ retrieves detailed information about a selected feature in the map (e.g., the pop up in Figure 1). The request is triggered by clicking an object on the map.

Logical Tier. The logical tier (server) accepts requests via a Java Servlet. An isochrone request invokes MINEX to compute an isochrone that is represented as a logical network. This representation is passed to the *GeoBuilder* module, which performs two tasks. First, the isochrone is annotated with geometry information and stored in vector format in a spatial relation in the DB. Second, the network representation of the isochrone is transformed into a spatial area (polygon). In order to enable the client to access the isochrone via WMS and WFS, the three tables (vertices, edges, and areas) are registered as vector layers in the map builder module. As Map Builder we use the rendering engine *Geoserver*[10], which is accessible via standardized OGC services. For a WMS request, Geoserver reads the spatial data from the DB and creates an image that is sent back to the client. For a WFS request, the information is retrieved from the DB and sent to the client as a feature in text format. The Map Builder serves also as proxy for providing base layers from external servers (e.g., Google, OpenStreetMap, Bing, etc.). The second main task of the logical tier is to support geospatial reachability analysis. Once an isochrone is computed, the user can specify an arbitrary SQL query that is sent to the server via a *geoAnalysis request* ④. The result of this SQL query is a table of geo-referenced objects that are joined with the isochrone, e.g., the apartments in Figure 1. The final result is converted from a relational format into a JSON object and sent back to the client.

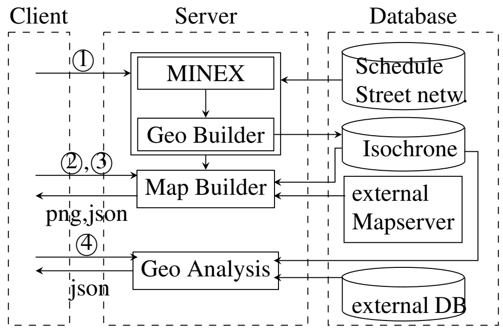


Fig. 5. Architecture

Data Tier. The data tier uses a relational DBMS with a spatial extension to perform spatial operations, such as edge clipping if an edge is only partially reached, locating the query point to the closest edge, area buffering, or spatial intersection. The ISOGA

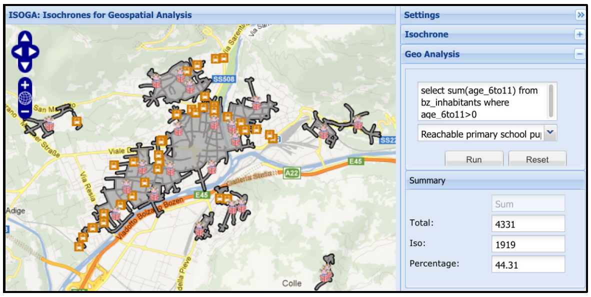
[9] `http://www.opengeospatial.org/` [10] `http://www.geoserver.org`

system works with PostGIS 2.0 as well with Oracle11g. Only standardized spatial operators (OGC/SQL-MM) are used, which simplifies the migration to other spatial DBMSs.

5 Application Scenarios

In this section we describe two application scenarios that emerged from a collaboration with the local municipality and one example that shows network expiration to illustrate the low memory requirements of MINEX.

Scenario 1. We want to determine how well the primary schools in Bozen-Bolzano are reachable by walking and taking the public transport system (Figure 6). For this we compute an isochrone with multiple query points that represent the schools, a maximal duration of 15 minutes, and an arrival time 9 am at the schools. Next, we specify an SQL query that retrieves from the inhabitants database all kids with an age between 6 and 11 years. The result of the SQL query is joined with the isochrone. The statistics shows the number and percentage of kids that reach the closest school in less than 15 minutes. By selecting the option *Outside Isochrone* for the join, it is possible to identify the number of kids who do not reach the school within the given time constraints.



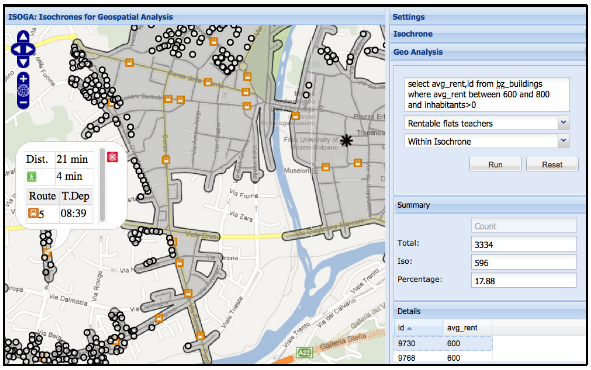
```
select sum(age_6to11) from
bz_inhabitants where
age_6to11>0
```

Reachable primary school pu

	Sum
Total:	4331
Iso:	1919
Percentage:	44.31

Fig. 6. Reachability of Primary Schools

Scenario 2. We want to determine cheap apartments that are close to the working place (Figure 7). The user specifies a query point on the map that represents his/her working place, a maximal acceptable traveling duration, and a price range he/she is willing to pay. After computing the isochrone, an SQL query retrieves all flats in the specified price range. The result is joined with the isochrone and all available flats that are located in the isochrone are visualized as circles on the map. The example shows additional information for one such apartment, namely that the working place can be reached in 25 min by walking first for 4 min to the closest bus station, where to take bus line 5 at 08:39 am.



Dist.	21 min
🚶	4 min
Route	T.Dep
5	08:39

```
select avg_rent,id from bz_buildings
where avg_rent between 600 and 800
and inhabitants>0
```

Rentable flats teachers

Within Isochrone

	Count
Total:	3334
Iso:	596
Percentage:	17.88

id ▲	avg_rent
9730	600
9768	600

Fig. 7. Flat Search Scenario

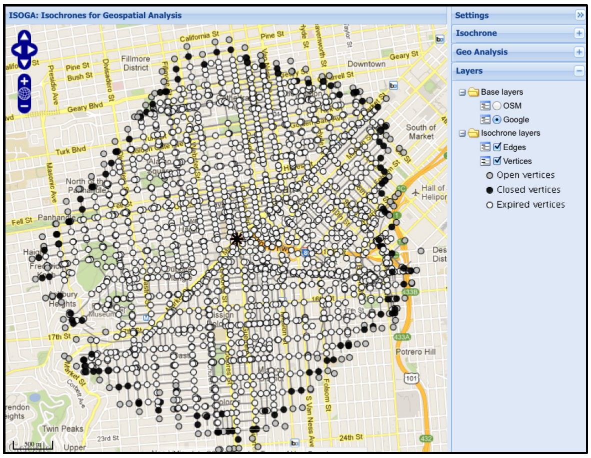


Fig. 8. Vertex Expiration in MINEX

Scenario 3. In this scenario we illustrate the low memory consumption of MINEX due to network expiration (Figure 8). The user can select different datasets. Currently, the cities of Bolzano-Bozen, Washington DC, and San Francisco as well as the regional networks of South Tyrol and Italy are available, all having different network topologies and transportation systems. After computing an isochrone, the user can open the Layers panel and activate the visualization of vertex expiration. Black circles represent the vertices that are kept in memory to avoid cyclic expansions. White circles represent expired vertices that are removed from memory to minimize the memory requirements.

6 Conclusion and Future Work

In this paper, we presented the web-based system ISOGA that uses isochrones to perform geospatial reachability analysis. Core features of the system are the support of multimodal networks and the possibility to join isochrones with the result of general SQL queries, which allows to analyze the reachability of various types of geo-referenced objects without the need to compute the distance of each object. The system adopts a service-oriented, three-tier architecture and uses technologies that are compliant with OGC standards. Future work includes the further development of the analysis component as well as the implementation of ISOGA for a mobile client.

References

1. Bauer, V., Gamper, J., Loperfido, R., Profanter, S., Putzer, S., Timko, I.: Computing isochrones in multi-modal, schedule-based transport networks. In: GIS, pp. 1–2. ACM (2008)
2. Deng, K., Zhou, X., Shen, H.T., Sadiq, S.W., Li, X.: Instance optimal query processing in spatial networks. VLDB J. 18(3), 675–693 (2009)
3. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
4. Gamper, J., Böhlen, M.H., Cometti, W., Innerebner, M.: Defining isochrones in multimodal spatial networks. In: CIKM, pp. 2381–2384 (2011)
5. Gamper, J., Böhlen, M., Innerebner, M.: Scalable Computation of Isochrones with Network Expiration. In: Ailamaki, A., Bowers, S. (eds.) SSDBM 2012. LNCS, vol. 7338, pp. 526–543. Springer, Heidelberg (2012)

6. Innerebner, M., Böhlen, M.H., Timko, I.: A web-enabled extension of a spatio-temporal dbms. In: GIS, pp. 34–41. ACM (2007)
7. Marciuska, S., Gamper, J.: Determining Objects within Isochrones in Spatial Network Databases. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 392–405. Springer, Heidelberg (2010)
8. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB, pp. 802–813 (2003)