Computing Isochrones in Multi-Modal, Schedule-Based Transport Networks*

Veronika Bauer
Free University of Bolzano
veronika.bauer@unibz.it

Johann Gamper
Free University of Bolzano
gamper@inf.unibz.it

Roberto Loperfido
Municipality of Bolzano
roberto.loperfido@comune.bz.it

Sylvia Profanter
Municipality of Bolzano
sylvia.profanter@comune.bz.it

Stefan Putzer
CreaForm
s.putzer@planalpin.it

Igor Timko
Free University of Bolzano
timko@inf.unibz.it

ABSTRACT

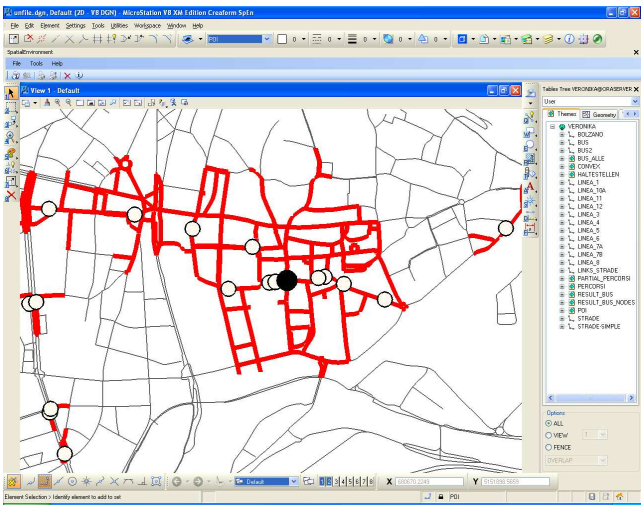
Isochrones are defined as the set of all points from which a specific point of interest is reachable within a given time span. This demo paper presents a solution to compute isochrones in multi-modal, schedule-based transport networks. The system is implemented in Java on top of the Oracle Spatial Network Model and is deployed at the Municipality of Bolzano-Bozen as a planning instrument.

1. INTRODUCTION

In urban planning, it is important to assess the coverage of the city area by various kinds of public services. An effective way to do so is to compute isochrones. An *isochrone* for a given point of interest (POI) is generally defined as the set of all points on a road network from which that POI can be reached in a specific time span. The computation of isochrones becomes more complicated when considering *multi-modal*, *schedule-based* transport networks, as the result depends not only on the time span, but due to varying schedules also on the selected date and arrival time at the POI.

This paper presents a system to compute isochrones in multi-modal, schedule-based networks, where walking in combination with the public transport is considered. Figure 1 shows a screenshot of the system with the 15-minutes isochrone at 11:25 for a single POI (black circle). The isochrone is represented as the collection of all street segments (bold gray lines in Fig. 1), from which the POI can be reached when starting at 11:10 (or later) and arriving at the POI no later than 11:25. There is a large "island" around the POI, which is within walking distance of 15 minutes. Smaller "islands" are around the bus stops (white circles in Fig. 1), from where the POI can be reached by a combination of going by foot and by bus. To adapt for different user groups such as elderly people and children, the walking speed can be specified as an additional parameter, e.g., 0.6 m/s in the above example.

In the field of databases, previous research work concentrated on queries in unimodal transport networks with fixed travel times (e.g. [1, 5]). Only some recent work makes more realistic assumptions and assumes time-dependent travel times — changing traffic situations in [3, 6] and schedules in [4] — but they all focus on the conventional shortest-path problem. Instead, our work is about isochrone computation in multi-modal networks that consider going by foot and public transport such as buses. In such a setup, the travel times vary depending on the bus schedules, and the possibility to change the bus has to be considered. Transportation engineers have also recently turned their attention to the schedule-based modeling of multi-modal transportation [8], but they don't consider implementations on top of DBMSes.



1: Screenshot of Isochrone Calculation.

To summarize, the contributions of this work are as follows:

- the design of an algorithm for computing isochrones in multi-modal, schedule-based transport networks, including the design of database tables for representing bus schedules;
- the implementation of the algorithm on top of the Oracle Spatial Network Model and its integration in a system for spatial data management.

2. THE ISOCHRONE ALGORITHM

Since our algorithm is based on a network structure that consists of nodes connected by links, we will subsequently use these terms. We assume one network that stores the streets of the city, where the links represent street segments (typically between two crossroads and/or joins), and a separate network for each bus line, where a link (logically) represents a leg between two consecutive bus stops. The networks are connected by a mapping table which maps each bus stop to the corresponding link in the street network (typically

*This work has been done in the context of the eBZ project, which is funded by the Municipality of Bolzano-Bozen.

ACM GIS'08, November 5–7, 2008, Irvine, CA, USA.

the bus stops do not coincide with nodes in the street network, but are located somewhere on the links). Algorithm 1 shows a skeleton of the basic algorithm for the computation of isochrones. The input parameters are: a set of networks, N; the destination node (POI), n_d (which is a node in the street network); the travel duration, dur; the arrival time, t_d; and the walking speed in m/s, s_w. The algorithm returns the subset of links from the street network, $N_{streets}$, that form the specified isochrone.

Algorithm: ISOCHRONE(N, n_d, t_d, dur, s_w)
Add $(n_d, t_d, N_{streets})$ to empty queue Q;
while $Q \neq \emptyset$ **do**
  $(n, t, N) \leftarrow$ next node from Q;
  foreach *(incoming) link* $(n', n) \in N$ **do**
    Compute latest starting time, t', for n';
    if $t' > t_d - dur$ **then**
      Add (n', t', N) to Q;
      if $N = N_{streets}$ **then**
        Add (n', n) to result set;
        Check for bus stops on (n', n) and add them to Q with appropriate starting time;
      else if N *is a bus network* **then**
        Check for stops of other buses at n' and add them to Q with appropriate starting time;

1: Algorithm for Isochrone Computation.

The algorithm is doing an exhaustive, backward search starting from the POI and traversing all paths until all links are found from which the POI can be reached in time. The queue, Q, maintains the nodes that have to be explored. Each node in the queue has associated (i) the latest starting time from that node in order to reach the POI in time and (ii) the network it belongs to. The queue is intialized with the node that represents the POI, associated with the arrival time, t_d. Then the nodes in the queue are processed in an iterative way, expanding one node at a time. When expanding a node in a bus network, only incoming links are considered, while for nodes in the street network the link direction is not relevant. Whenever a node, n', is reached that is still within the time limit, it is added to Q with a new latest starting time, t'. To compute t', the walking speed is considered if the link belongs to the street network, and the schedule is queried if the link belongs to a bus line. To correctly manage the integration of all networks — that is to allow the switching between walking and using buses as well as changing buses — we use a mapping table and check whether bus stops are located on the corresponding street segment or whether a change to another bus line is possible.

3. IMPLEMENTATION

The isochrone algorithm has been implemented on top of the Oracle Spatial Network Model [10] and has been integrated into Spatial Environment, an expressive and flexible geodata editing and publishing environment developed by CreaForm [7, 9] that is used at the Municipality of Bolzano-Bozen. In Oracle Spatial, a network is represented as a (possibly directed) graph consisting of nodes and links stored in the two standard tables <*NETWORK*>*_NODE$* and <*NETWORK*>*_LINK$*, where <*NETWORK*> is the name of the network. Table 1 shows an excerpt of the link table for the street network and the network for bus line 10. The representation of the bus lines is more complicated, since the leg from one bus stop to the next is typically composed of several links, following the shape of the underlying street network. Therefore, the built-in <*NETWORK*>*_PATH$* table is used to provide a logical view that represents a bus line by a single link between any two consecutive

(a)

Link	Start_Node	End_Node	Cost	Bidir	Geometry
2101	1604	1483	296	Y	xxx
2121	1840	1395	54	Y	xxx

(b)

Link	Start_Node	End_Node	Bidir	Goemetry
1	5050	5048	N	xxx
30	5229	5181	N	xxx
31	5181	5183	N	xxx
32	5183	5185	N	xxx

1: Link Tables: (a) Street Network and (b) Bus Line 10A

stops (i.e., linking only those nodes that represent also a stop).

As we have noticed, bus stops typically do not coincide with a node in the underlying street network, but are located on a street link. Table 2 shows an excerpt from the mapping table that connects bus stops with points on the street network. Each table entry represents a bus stop and its position in the street network, expressed as the corresponding street link plus an offset from that link's start node and end node, respectively.

BUS_STOP	STREET_LINK	START_NODE	END_NODE	START_DISTANCE	END_DISTANCE
5166	2101	1604	1483	285	11
5233	2121	1840	1395	6	48

2: Mapping Table

For storing bus schedules we separate the bus route and its timetable (see Table 3). A bus schedule is then represented as a set of individual trips that are represented by a trip id, a day marker, arrival time at each bus stop, etc.

(a)

BUS_LINE	STOP_ID	POSITION
LINEA_10A	5156	1
LINEA_10A	5171	2
LINEA_10A	5233	3

(b)

STOP_ID	TRIP_ID	MARKER	BUS_LINE	ARRIVAL
5156	14087	WEEK	LINEA_10A	10:22
5171	14087	WEEK	LINEA_10A	10:23
5233	14087	WEEK	LINEA_10A	10:24

3: Bus Schedules: (a) Route Table and (b) Time Table.

4. CONCLUSIONS AND FUTURE WORK

This paper presents a system for the computation of isochrones in multi-modal, schedule-based transport networks, where walking in combination with public transport is considered. The system computes the isochrones as the set of all street segments from which a given POI can be reached within the given time interval. Further input parameters are the desired arrival time and the walking speed. The system is implemented in Java on top of Oracle Spatial and it is deployed at the Municipality of Bolzano-Bozen.

Future work includes the modeling of advanced bus schedules, e.g, with holidays and weekend bus trips. Furthermore, we will improve the basic algorithm in various directions: allow for partial segments at the border of the isochrone, which is important if the street segments are rather long; improved strategies to select the next point from Q to make the algoritm more efficient; consider other transportation modes such as bicycle; compute the isochrones as the concave hull that includes the street segments.

Acknowledgements

We would like to thank Gytis Tumas (Free University of Bolzano) for his contribution into this paper.

5. REFERENCES

[1] V. T. de Almeida and R.H. Güting. Using Dijkstra's Algorithm to Incrementally Find the K-Nearest Neighbors in Spatial Network Databases. In *Proc. SAC*, 2006.

[2] E. W.Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1: 269–271, 1959.

[3] B. Ding et al. Finding Time-Dependent Shortest Paths over Large Graphs. In *Proc. EDBT*, pp. 205–215, 2008.

[4] R. Huang. A Schedule-based Pathfinding Algorithm for Transit Networks Using Pattern First Search. *Geoinformatica* 11:269–285, 2007.

[5] C. S. Jensen et al. Nearest Neighbor Queries in Road Networks. In *Proc. ACM GIS*, pp.1–8, 2003.

[6] E. Kaboulas et al. Finding Fastest Paths on a Road Network with Speed Patterns. In *Proc. ICDE*, 2006.

[7] S. Putzer and G. Lavoriero. Complete eGovernment solution at City of Bozen-Bolzano. 2008 Oracle Spatial User Conference.

[8] N. H. M. Wilson and A. Nuzzolo (ed.). Schedule-Based Dynamic Transit Modeling. *Kluwer Academic Publishers*, 2004.

[9] Creaform, Italy. http://www.creaform.com.

[10] Oracle Spatial Network Data Model. *An Oracle Technical White Paper*. http://www.oracle.com/technology/products/spatial/pdf/10gr2_collateral/spatial_twp_ntwrkdatamod_10gr2_0512.pdf. May 2005.