# Incremental calculation of isochrones regarding duration

Nikolaus Krismer
University of Innsbruck, Austria
nikolaus.krismer@uibk.ac.at

Günther Specht
University of Innsbruck, Austria
guenther.specht@uibk.ac.at

Johann Gamper
Free University of Bozen-Bolzano, Italy
gamper@inf.unibz.it

## ABSTRACT

An isochrone in a spatial network is the minimal, possibly disconnected subgraph that covers all locations from where a query point is reachable within a given time span and by a given arrival time [5]. A novel approach for computing isochrones in multimodal spatial networks is presented in this paper. The basic idea of this incremental calculation is to reuse already computed isochrones when a new request with the same query point is sent, but with different duration. Some of the major challenges of the new calculation attempt are described and solutions to the most problematic ones are outlined on basis of the already established MINE and MINEX algorithms. The development of the incremental calculation is done by using six different cases of computation. Three of them apply to the MINEX algorithm, which uses a vertex expiration mechanism, and three cases to MINE without vertex expiration. Possible evaluations are also suggested to ensure the correctness of the incremental calculation. In the end some further tasks for future research are outlined.

## Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

## General Terms

Algorithms

## Keywords

isochrone, incremental calculation

## 1. INTRODUCTION

Throughout the past years interactive online maps have become a famous tool for planning routes of any kind. Nowadays everybody with access to the internet is able to easily get support when travelling from a given point to a specific target. The websites enabling such a navigation usually calculate routes using efficient shortest path (SP) algorithms. One of the most famous examples of these tools is Google's map service named GoogleMaps[1]. For a long time it was possible to calculate routes using one transportation system (by car, by train or by bus) only. This is known as routing within unimodal spatial networks. Recent developments enabled the computation combining various transportation systems within the same route, even if some systems are bound to schedules. This has become popular under the term "multimodal routing" (or routing in multimodal spatial networks).

Less famous, but algorithmic very interesting, is to find the answer to the question where someone can travel to in a given amount of time starting at a certain time from a given place. The result is known as isochrone. Within multimodal spatial networks it has been defined by Gamper et al. [5]. Websites using isochrones include Mapnificent[2] and SimpleFleet[3] [4].

One major advantage of isochrones is that they can be used for reachability analyses of any kind. They are helpful in various fields including city planning and emergency management. While some providers, like SimpleFleet and Mapnificent, enable the computation of isochrones based on pre-calculated information or with heuristic data, the calculation of isochrones is a non-trivial and time-intense task. Although some improvements to the algorithms that can be used for isochrone computation have been published at the Free University of Bozen-Bolzano in [7], one major drawback is that the task is always performed from scratch. It is not possible to create the result of a twenty-minute-isochrone (meaning that the travelling time from/to a query point `q` is less than or equal to twenty minutes) based on the result from a 15-minute-isochrone (the travelling time is often referred to as maximal duration `dmax`). The incremental calculation could dramatically speed up the computation of isochrones, if there are other ones for the same point `q` available. This is especially true for long travel times. However, the computation based on cached results has not been realised until now and is complex. As one could see from figures 1 and 2 it is not sufficient to extend the outline of the isochrone, because there might be some network hubs (e.g. stations of the public transportation system) which extend the isochrone result into new, possibly disconnected areas.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

[1]http://maps.google.com
[2]http://www.mapnificent.net
[3]http://www.simplefleet.eu

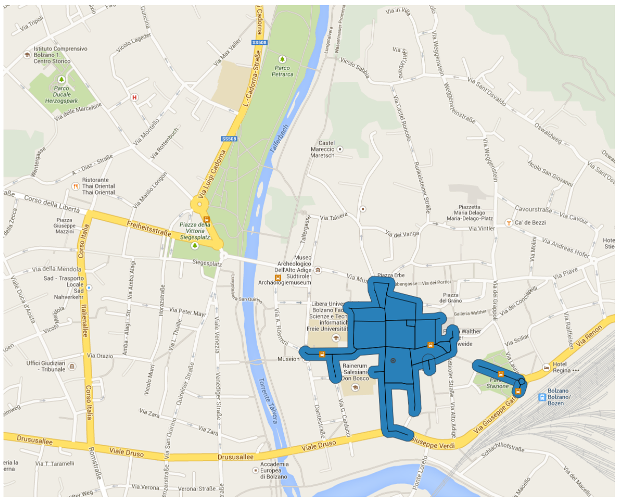

**Figure 1: Isochrone with dmax of 10 minutes**

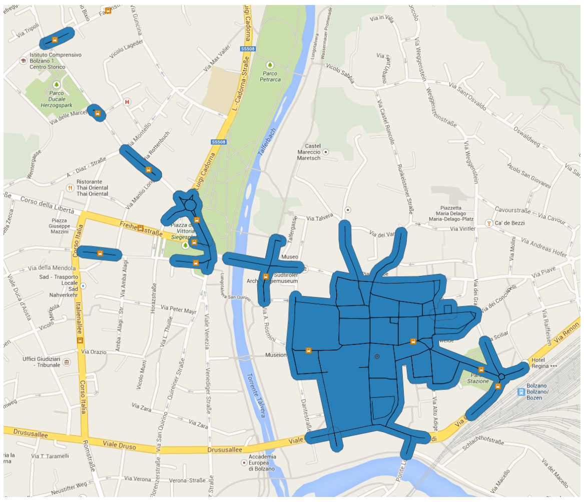

**Figure 2: Isochrone with dmax of 15 minutes**

This paper presents the calculation of incremental isochrones in multimodal spatial networks on top of already developed algorithms and cached results. It illustrates some ideas that need to be addressed when extending the algorithms by the incremental calculation approach. The remainder of this paper is structured as follows. Section 2 includes related work. Section 3 is split into three parts: the first part describes challenges that will have to be faced during the implementation of incremental isochrones. Possible solutions to the outlined problems are also discussed shortly here. The second part deals with different cases that are regarded during computation and how these cases differ, while the third part points out some evaluations and tests that will have to be performed to ensure the correctness of the implementation. Section 4 consists of a conclusion and lists some possible future work.

## 2. RELATED WORK

The calculation of isochrones in multimodal spatial networks can be done using various algorithms. The method introduced by Bauer et. al [1] suffers from high initial loading time and is limited by the available memory, since the entire network is loaded into memory at the beginning. Another algorithm, called Multimodal Incremental Network Expansion (MINE), which has been proposed by Gamper et al. [5] overcomes the limitation that the whole network has to be loaded, but is restricted to the size of the isochrone result, since all points in the isochrone are still located in memory. To overcome this limitation, the Multimodal Incremental Network Expansion with vertex eXpiration (MINEX) algorithm has been developed by Gamper et al. [6] introducing vertex expiration (also called node expiration). This mechanism eliminates unnecessary nodes from memory as soon as possible and therefore reduces the memory needed during computation.

There are some more routing algorithms that do not load the entire network into the main memory. One well-known, which is not specific for isochrone calculation, but for query processing in spatial networks in general, called Incremental Euclidean Restriction (IER), has been introduced by Papadias [8] in 2003. This algorithm loads chunks of the network into memory that are specified by the euclidean distance. The Incremental Network Expiration (INE) algorithm has also been introduced in the publication of Papadias. It is basically an extension of the Dijkstra shortest path algorithm. Deng et al. [3] improved the ideas of Papadias et al. accessing less network data to perform the calculations. The open source routing software "pgRouting"[4], which calculates routes on top of the spatial database PostGIS[5] (an extension to the well-known relational database PostgreSQL) uses an approach similar to IER. Instead of the euclidean distance it uses the network distance to load the spatial network.

In 2013 similar ideas have been applied to MINEX and resulted in an algorithm called Multimodal Range Network Expansion (MRNEX). It has been developed at the Free University of Bozen-Bolzano by Innerebner [7]. Instead of loading the needed data edge-by-edge from the network, it is loaded using chunks, like it is done in IER. Depending on their size this approach is able to reduce the number of network accesses by far and therefore reduces calculation time.

Recently the term "Optimal location queries" has been proposed by some researches like Chen et al. [2]. These queries are closely related to isochrones, since they "find a location for setting up a new server such that the maximum cost of clients being served by the servers (including the new server) is minimized".

## 3. INCREMENTAL CALCULATION REGARDING ISOCHRONE DURATION

In this paper the MINE and MINEX algorithms are extended by a new idea that is defined as "incremental calculation". This allows the creation of new results based on already computed and cached isochrones with different durations, but with the same query point q (defined as base-isochrones). This type of computation is complex, since it is not sufficient to extend an isochrone from its border points. In theory it is necessary to re-calculate the isochrone from every node in the spatial network that is part of the base-isochrone and connected to other nodes. Although this is

[4]http://pgrouting.org
[5]http://postgis.net

true for a highly connected spatial network it might not be the only or even best way for a real-world multimodal spatial network with various transportation systems. The isochrone calculation based on already known results should be doable with respect to all the isochrone's border points and all the public transportation system stations that are part of the base isochrone. These network hubs in reality are the only nodes, which can cause new, possibly disconnected areas to become part of an isochrone with different travelling time.

As it is important for the incremental calculation, the vertex expiration that is introduced by Gamper et al. in [6] will now be summarized shortly. The aim of the proposed approach is to remove loaded network nodes as soon as possible from memory. However, to keep performance high, nodes should never be double-loaded at any time and therefore they should not be eliminated from memory too soon. Removal should only occur when all computations regarding the node have been performed. States are assigned to every node to assist in finding the optimal timeslot for memory elimination. The state of a node can either be "open", "closed" or "expired". Every loaded node is labelled with the open state in the beginning. If all of its outgoing edges are traversed, its state changes to closed. However, the node itself has to be kept in memory in order to avoid cyclic network expansions. A node reaches the expired state, if all nodes in its neighbourhood reached the closed or expired state. It then can safely be removed from memory and is not available for further computations without reloading it from the network. Since this is problematic for the incremental calculation approach this aspect is described in more detail.

## 3.1 Challenges

There are some challenges that need to be addressed when implementing an incremental calculation for the MINE and MINEX algorithm. The most obvious problem is related to the vertex expiration of the MINEX algorithm. If nodes already expired, they will not be available to the calculation of isochrones with different durations. To take care of this problem all nodes `n` that are connected to other nodes which are not in the direct neighbourhood of `n` are added to a list `l_hubs`. These nodes are the ones we referred to as network hubs. Besides the hub node `n` itself, further information is stored in this list: the time `t` of arrival at the node and the remaining distance `d` that can be used. With this information it is possible to continue computation from any hub with a modified travelling time for the algorithms.

The list `l_hubs` needs to be stored in addition to the isochrone's maximal travelling time and the isochrone result itself, so that it can be used for incremental calculation. None of this information needs to be held in memory during computation of the base isochrone itself and is only touched on incremental calculation. Therefore, runtime and memory consumption of the isochrone algorithms will not be influenced much.

Other problems include modifications to the spatial network in combination with incremental isochrones. If there is some change applied to the underlying network, all the base isochrones can not be used for incremental calculation any more. It can not be guaranteed that the network modification does not influence the base isochrone. Changes in the schedules of one or more modalities (for example the public transportation systems) could cause problems as well, as they would also influence the base isochrone. Schedule alternations can be triggered by a service provider. Traffic jams and similar factors can lead to delays in the transportation system and thus also have to be considered. Although it should be possible to overcome both limitations or at least limit their impact, it will not be further discussed in this paper.

## 3.2 Types of calculation

There are six different cases that have to be kept in mind when calculating an isochrone with travelling time `dmax` using a base isochrone with duration `dmax_base`: three applying to algorithms without vertex expiration and three cases for the ones using vertex expiration.

### *3.2.1 Cases* `dmax = dmax_base`

The first two and most simple cases for the MINE and MINEX algorithm, are the ones where `dmax` is equal to `dmax_base`. In these cases it is obvious that the calculation result can be returned directly without any further modification. It is not needed to respect expired nodes, since no (re)calculation needs to be performed.

### *3.2.2 Cases* `dmax < dmax_base`

The third, also simple case, is the one where `dmax` is less than `dmax_base` for algorithms without vertex expiration. In this situation all nodes can be iterated and checked for suitability. If the duration is less or equal to `dmax`, then the node also belongs to the new result, otherwise it does not. In the fourth case, where the duration is less than `dmax_base` and nodes were expired (and therefore are not available in memory any more), the isochrone can be shrunk from its borders. The network hubs do not need any special treatment, since no new areas can become part of the result if the available time decreased. The only necessary task is the recalculation of the durations from the query point to the nodes in the isochrone and to possibly reload expired nodes. It either can be done from the query point or from the border points. The duration `d` from the query point `q` to a network node `n` is then equal to (assuming that the border point with the minimal distance to `n` is named `bp`):

$$d(q, n) = d(q, bp) - d(bp, n)$$

### *3.2.3 Cases* `dmax > dmax_base`

The remaining two cases, where `dmax_base` is less than `dmax`, are much more complex. They differ in the fact that new, possibly disconnected areas can become part of the result and therefore it is not sufficient to look at all the base isochrones border points. The new areas become available as a result from connections caused by network hubs that often are bound to some kind of schedule. A real-world example is a train station where a train is leaving at time `t_train` due to its schedule and arriving at a remote station at or before time `dmax` (in fact any time later than `dmax_base` is feasible). The time `t_train` has to be later than the arrival time at the station (and after the isochrones starting time).

Since all network hubs are saved with all the needed information to the list `l_hubs` it is not of any interest if the algorithm uses vertex expiration or not. The points located at the isochrone's outline are still in memory. Since only network hubs can create new isochrone areas it is sufficient to grow the isochrone from its border and all the network hubs located in the isochrone. The only effect that vertex expiration causes is a smaller memory footprint of the calculation, as it would also do without incremental calculation.

In table 1 and in table 2 the recently mentioned calculation types are summarised shortly. The six different cases can be distinguished with ease using these two tables.

| | MINE |
|---|---|
| `dmax` < `dmax_base` | iterating nodes from base isochrone checking if travel time is <= `dmax` |
| `dmax` = `dmax_base` | no change |
| `dmax` > `dmax_base` | extend base isochrone by border points and with list `l_hubs` |

**Table 1:**
**Incremental calculation without vertex expiration**

| | MINEX |
|---|---|
| `dmax` < `dmax_base` | shrink base isochrone from border |
| `dmax` = `dmax_base` | no change |
| `dmax` > `dmax_base` | extend base isochrone by border points and with list `l_hubs` |

**Table 2:**
**Incremental calculation with vertex expiration**

Although the different types of computations are introduced using the MINE and MINEX algorithms they also apply to the MRNEX method. When using MRNEX the same basic idea can be used to enable incremental calculations. In addition the same advantages and disadvantages apply to the incremental calculation using MRNEX compared to MINEX that also apply to the non-incremental setup.

### 3.3 Evaluation

The evaluations that will need to be carried out to ensure the correctness of the implementation can be based on freely available datasets, such as OpenStreetMap[6]. Schedules from various public transportation systems could be used and since they might be subject of licensing it is planned to create some test schedules. This data can then be used as mockups and as a replacement of the license-bound real-world schedules. It is also planned to realise all the described tests in the context of a continuous integration setup. They will therefore be automatically executed ensuring the correctness throughout various software changes.

The basic idea of the evaluation is to calculate incremental isochrones on the basis of isochrones with different durations and to compare them with isochrones calculated without the incremental approach. If both results are exactly the same, the incremental calculation can be regarded as correct.

There will be various tests that need to be executed in order to cover all the different cases described in section 3.2. As such, all the cases will be performed with and without vertex expiration. The durations of the base isochrones will cover the three cases per algorithm (less than, equal to and greater than the duration of the incremental calculated isochrone). Additional tests, such as testing for vertex expiration of the incremental calculation result, will be implemented as well. Furthermore, the calculation times of both - the incremental and the non-incremental approach - will be recorded to allow comparison. The incremental calculation can only be seen as successful, if there are situations where they perform better than the common calculation. As mentioned before, this is expected to be true for at least large isochrone durations, since large portions of the spatial network does not need to be loaded then.

[6]http://www.openstreetmap.org

Besides these automatically executed tests, it will be possible to perform manual tests using a graphical user interface. This system is under heavy development at the moment and has been named IsoMap. Regardless of its young state it will enable any user to calculate isochrones with and without the incremental approach and to visually compare the results with each other.

## 4. CONCLUSION AND FUTURE WORK

In this paper an approach to enable the calculation of isochrones with the help of already known results was presented. The necessary steps will be realised in the near future, so that runtime comparisons between incremental calculated isochrones and such created without the presented approach will be available shortly. The ideas developed throughout this paper do not influence the time needed for calculation of base isochrones by far. The only additional complexity is generated by storing a list `l_hubs` besides the base isochrone. However, this is easy to manage and since the list does not contain any complex data structures, the changes should be doable without any noticeable consequence to the runtime of the algorithms.

Future work will extend the incremental procedure to further calculation parameters, especially to the arrival time, the travelling speed and the query point q of the isochrone. Computations on top of cached results are also realisable for changing arrival times and/or travel speeds. It should even be possible to use base isochrones with completely different query points in the context of the incremental approach. If the isochrone calculation for a duration of twenty minutes reaches a point after five minutes the 15-minute isochrone of this point has to be part of the computed result (if the arrival times are respected). Therefore, cached results can decrease the algorithm runtimes even for different query points, especially if they are calculated for points that can cause complex calculations like airports or train stations.

Open fields that could be addressed include the research of incremental calculation under conditions where public transportation system schedules may vary due to trouble in the traffic system. The influence of changes in the underlying spatial networks to the incremental procedure could also be part of future research. It is planned to use the incremental calculation approach to calculate city round trips and to allow the creation of sight seeing tours for tourists with the help of isochrones. This computation will soon be enabled in cities where it is not possible by now.

Further improvements regarding the calculation runtime of isochrones can be done as well. In this field, some examinations with different databases and even with different types of databases (in particular graph databases and other NoSQL systems) are planned.

## 5. REFERENCES

[1] V. Bauer, J. Gamper, R. Loperfido, S. Profanter, S. Putzer, and I. Timko. Computing isochrones in multi-modal, schedule-based transport networks. In

*Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 78:1–78:2, New York, NY, USA, 2008. ACM.

[2] Z. Chen, Y. Liu, R. C.-W. Wong, J. Xiong, G. Mai, and C. Long. Efficient algorithms for optimal location queries in road networks. In *SIGMOD Conference*, pages 123–134, 2014.

[3] K. Deng, X. Zhou, H. Shen, S. Sadiq, and X. Li. Instance optimal query processing in spatial networks. *The VLDB Journal*, 18(3):675–693, 2009.

[4] A. Efentakis, N. Grivas, G. Lamprianidis, G. Magenschab, and D. Pfoser. Isochrones, traffic and demographics. In *SIGSPATIAL/GIS*, pages 538–541, 2013.

[5] J. Gamper, M. Böhlen, W. Cometti, and M. Innerebner. Defining isochrones in multimodal spatial networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2381–2384, New York, NY, USA, 2011. ACM.

[6] J. Gamper, M. Böhlen, and M. Innerebner. Scalable computation of isochrones with network expiration. In A. Ailamaki and S. Bowers, editors, *Scientific and Statistical Database Management*, volume 7338 of *Lecture Notes in Computer Science*, pages 526–543. Springer Berlin Heidelberg, 2012.

[7] M. Innerebner. *Isochrone in Multimodal Spatial Networks.* PhD thesis, Free University of Bozen-Bolzano, 2013.

[8] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 802–813. VLDB Endowment, 2003.