What Algorithms for Urban Routing on Mobile Devices?

Tristram Gräbener, Alain Berro, Yves Duthen
IRIT-UT1, UMR 5505, CNRS
Université de Toulouse
2 rue du doyen Gabriel Marty 31042 Toulouse Cedex 9, France
{tristram.grabener,alain.berro,yves.duthen}@irit.fr

Abstract: This article presents a model for the multimodal shortest path problem. That model allows to use existing shortest path algorithm implementation and thus is very efficient and optimal. The performances are well suited for a mobile device like a smartphone.

We also present how multiobjective optimization improves the user interface by presenting similar solutions without any user interaction.

Finally we present the architecture of our prototype and give some experimental results.

Introduction

Today many mobile devices offer a GPS and a navigation system. However, those navigation systems are mostly oriented for car drivers: when a pedestrian mode exists, it is limited to the road network. It would be much more interesting to provide a complete itinerary to the destination including public transportation and possibly others means of transports like rental bikes. This problem is called multimodal shortest path.

However, such a system would be difficult to use as the user preferences are difficult to model and annoying to input on a mobile device: the best itinerary will depend on the user's personality, the weather, the luggage he might carry etc. Therefore presenting multiple solutions will allow the user to choose the most suited path with no need to enter the preferences.

We present in the first section a model for multimodal path calculation and the situations where good performances can be expected. The second sections presents multiobjective optimization and how it can greatly simplify the interaction on mobile devices. Then we present the architecture we developed for our prototype and some implementation details.

1 Multimodal transportation

Multimodal transportation consists in combining during a single itinerary multiple modes of transportation. An example would be to take the bike until the subway station, the subway and finally walk until the destination. Even if it becomes more and more popular among users and transportation network planing, only little research has been carried out about calculating optimal itineraries.

1.1 Graphs and paths

A valuated and oriented *graph* is a tuple $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{F})$ where $\mathcal{N} = \{1, 2, \ldots, n\}$ is the set of n nodes, $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ the set of $m = |\mathcal{E}|$ *edges* and $\mathcal{F} = \{F_{ij} \mid (i,j) \in \mathcal{E}\}$ the m cost functions associated to each edge.

A *path* $p = (p_i)_{i \in 1,\ldots,l}$ is an ordered sequences of $l = |p|$ nodes such that two successive nodes are connected by an edge $\forall i \in 1, \ldots, l-1, \quad (p_i, p_{i+1}) \in \mathcal{E}$. The cost of a path is calculated by applying the cost function of each arc $F(p) = F_{p_1 p_2} \circ F_{p_2 p_3} \circ \ldots \circ F_{p_{l-1} p_l}(t_0)$. Traditionally, an edge (i,j) has a constant weight w_{ij}. Hence the cost function is $F_{ij}(t) = w_{ij} + t$.

Modeling a road network as a graph is straightforward: the intersections are the nodes, and street sections are the edges. The weight on the edges can be length of the link or the travel time through that link.

Between a source s and a dwell t, out of all possible paths, the one that has smallest cost is called *shortest path*. Many algorithms were developed to find the shortest path. The most known is probably Dijkstra's that runs in $\mathcal{O}(n \cdot m)$ [Dij59]. It is proven that it finds the shortest path as long as cost functions are increasing. The formal proof and more general conditions of optimality can be found in [GM]. By selecting carefully the datastructures, it is possible to have the algorithm that runs in $\mathcal{O}(n \cdot \log n)$ [FT87].

1.2 Shortest path for public transport

Public transport are also modeled as a graph: every station is represented by a node and an edge exists if there is a line directly connecting two station. In the literature, to handle the waiting time at a station, it is usual to add a waiting time on every node to define it when searching for the shortest path. This requires to use specific algorithms as it can be more interesting to wait longer on a node (for example to catch a direct train instead of an omnibus). A graph is said to be *FIFO* (first in, first out) if there is no such situation: leaving earlier from a node guaranties to arrive earlier.

The figure1 depicts a non FIFO graph: to go the node 4, the fastest way is to use only the car, while to reach the node 3, the fastest way is to use the subway. Therefore it an algorithm like Dijkstra's cannot be used.

An other approach is to split the day into q time intervals and to expand the graph to a *time-space* graph. For every node of the original graph, q nodes a created. The edges in the time-space graph exist if it is possible to go from a node to an other if there is a connection that matches the time interval of both nodes. The *Chrono-SPT* algorithm presented in [PS98] calculates the shortest path in $\mathcal{O}(q \cdot m)$. The strongest aspect of this algorithm is

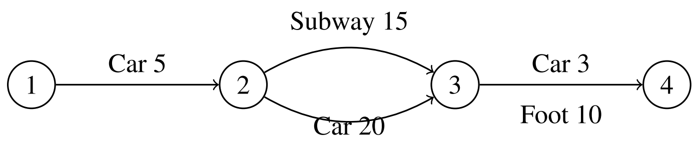

Figure 1: Non FIFO situation

to be able to implicitly handle the time-space graph and is quite memory efficient.

Both [ZW00] used that approach to solve the multimodal shortest path problem. This approach allowed them to solve the problem with 100 time intervals and 1000 nodes in a few seconds. A city like Toulouse has about 10000 nodes, and 100 time intervals represent 15 minutes.

As it is pointed out in [AOPS02], there is a significant difference between computing the *fastest* and the *minimum cost* shortest path in a dynamic network. The fastest path can be computed in polynomial time, while minimum cost is generally NP-hard. The intuition behind this property is that, as the cost depends on the time, it is necessary to explore every node at any time.

1.3 A model for multimodal shortest path

The model we present in this section allows to build a graph such that the FIFO constraint holds for the multimodal fastest path. With this transformation and by including the waiting time in the cost functions, our model allows the use of traditional algorithms and existing implementations.

One graph per transport mode Each transport mode is modeled by a different graph. The pedestrian graph and the bicycle graph will be very similar but the cost functions will be different. Also two trains using the same tracks, will be modeled as two different graphs if they don't mark the same stops. The nodes of two transport modes are connected if it is physically possible to switch from one mode to an other. The cost function on that connexion edge reflects the time needed to switch. There will be an edge in both directions if it is possible to switch, for example, from foot to public transportation and vice versa. By adding edges from the bike layer to the foot layer, but the not opposite, we model the fact that it is possible to start the itinerary by bike and once it is left somewhere, it won't be possible to use it again. The figure2 represents such a graph.

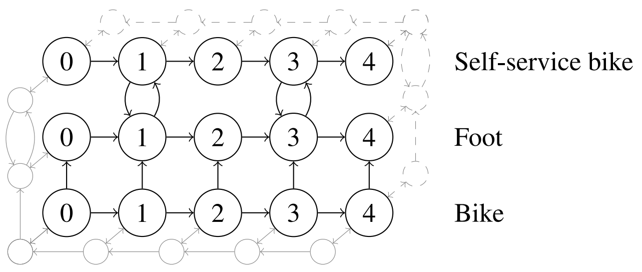

Figure 2: Multiple layer model

Including waiting time in cost function Instead of using separate waiting cost, we integrate the waiting time in the cost function. Given an edge $e = (i, j)$, the cost function would be $F_e(t) = w_e(t) + d_e$ where d_e is the duration on that edge (duration between two bus stops) and $w_e(t)$ the waiting time at the instant t until the next bus on the stop i. Considering that between two consecutive stops a bus won't overtake the previous one, this graph is FIFO. As the cost functions are increasing, it is possible to use the Dijkstra algorithm to compute the shortest path.

If the execution of the cost function is in constant time, then the global complexity of Dijkstra's algorithm will remain $\mathcal{O}(n \cdot \log n)$.

Discussion This model allows a very versatile and precise modeling of any transport mode. It allows to use a standard algorithms and thus use existing optimized implementations. First benchmarks prove the excellent performances of that approach.

However, this models requires more memory than a compact representation. The road map of the urban region of Toulouse (about one million inhabitants) requires 100Mb of memory. Therefore this isn't a real problem. The most restrictive condition is that only travel time can be optimized. Indeed any other objective depends on time (is a bus available? What will traffic be?) and thus breaking the FIFO constraint.

As it is impossible to ignore time, in order to handle other objectives at least two objectives have to be taken in account, hence multiobjective optimization cannot be avoided.

2 Multiobjective optimization

2.1 Benefits for a better user interface

Someone on the street with a mobile phone that is looking for the fastest itinerary will not want to spend time setting his preferences. Furthermore setting the preferences on a mobile device can be easily annoying. Last not least, it is not possible to save the preferences as they will be different on every use: if it is raining or if the user has heavy luggages, he would prefer avoiding multiple changes and long walks; on the contrary, if the weather is nice and he has some free time, he could enjoy walking even if it isn't the fastest way.

The objectives that are the most likely to be optimized are: time, cost, pollution (for example CO_2 emissions), number of changes and comfort. It is not possible to optimize each objective separately: the cheapest and less polluting will always be walking, but probably not the preference of the user. Aggregating the objectives into a single one is not desirable as it requires a precise configuration of the user.

Multiobjective optimization tries to optimize simultaneously all objective. The immediate consequence is that there is not one optimal solution, but a set of equivalent solutions: no solution beats any other solution on *all* objectives. This set is called the *Pareto front*.

Once the Pareto front is calculated, all solutions can be suggested to the user. According to his preferences, the user will choose the ideal itinerary.

Multiobjective optimization is a great tool to make interaction easier between a navigation system and the user as only the destination has to be entered. However calculating the Pareto front is not trivial.

2.2 Algorithms

Because most multiobjective optimization problems are NP-difficult and because there is not one, but multiples optima, most algorithms are based on genetic algorithms. More generally a population based algorithm is well suited to model the multiplicity of the solutions. For a general presentation of this topic, the reader can refer to [CVVL02].

However genetic algorithm aren't well suited for the point-to-point shortest path problem. Indeed, the usual genetic algorithms operators (mutation and crossing) aren't very natural to adapt to shortest path. Most research on multiobjective shortest path has been carried on exact algorithms. Several exact approaches are compared in [GM01].

Some others metaheuristics have been adapted to solve multiobjective problems like the particle swarm optimization or the ant colony optimization ([CVVL02] and [ASG07].

Ant colony optimization has been used successfully on some multiobjective problems, but not yet (to our knowledge) to the multiobjective shortest path problem. We believe that it is an promising approach, as this metaheuristic is inspired by ants searching for shortest paths.

3 Implementation and architecture

We created a prototype to demonstrate that our approach allows very fast itinerary calculations that scales well on bigger networks. The prototype is a web page where the user only has to select the start, destination and start time. The calculated itinerary is shown on a map detailing the transport modes to use.

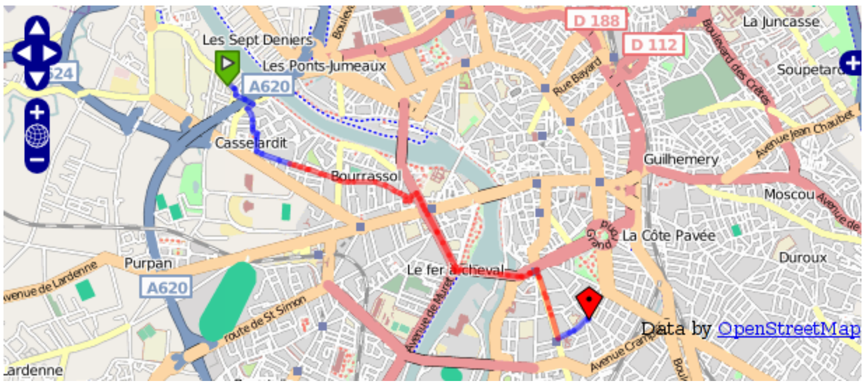

Figure 3: Screen shot of the prototype

3.1 Data

In our prototype we support pedestrian and self-service bikes in urban area of Toulouse. This bike rental system is available in most big cities in France and some other European cities: with a subscription it is possible to take and deposit a bike on any station. Toulouse has about 250 stations, and Paris about 1500.

The road network come from *OpenStreetMaps*[1] while the bike stations are available as XML files on the official website [2].

3.2 Implementation

On the client side (web-browser) we use OpenLayers, a javascript application, that allows to display the pre-rendered cartography and arbitrary paths.

On the server side, we use the implementation of Dijkstra's algorithm of the Boost Graph Library in C++. A Ruby on Rails application communicates with the client through Ajax. This allows an easy to use and reactive interface while minimizing the amount of data exchanged.

An example of the interface can be seen on figure3: a calculated itinerary using two different transport modes can be seen.

[1] `http://www.openstreetmap.org/`
[2] `http://www.velo.toulouse.fr/`

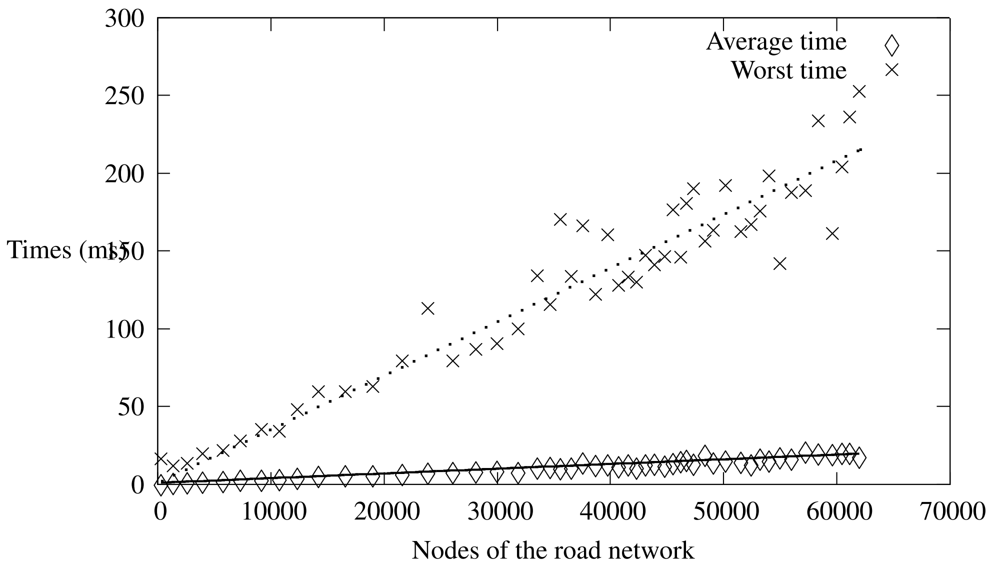

Figure 4: Multimodal shortest path computation time

3.3 Benchmarks

The benchmark were made by running 1000 itinerary calculations by choosing random start and destination nodes on 50 different graph sizes. This allows to test the average performance and how it scales. The tests were run on a 1.66GHz laptop.

The graphs are squares centered on Toulouse having a width between 2 a 100km, representing from 361 nodes up to 62020. The results are displayed on figure4. We can see that both the average and the worst running time out of 1000 calculations increase linearly with the number of nodes.

Those results are significantly better than in previous works. Therefor there is good hope to have an responsive software once we port our algorithm on a mobile device.

4 Future developments

The current models lacks of public transportations, and we hope to be able to include them very soon. We also believe that the model could easily be expanded to greater regions, like whole France. However this will require a bit of work in order to add heuristics that will limit the exploration space. Going on a even larger scale (the world for example) will be more problematic as it will not be possible to load all the data in memory.

While the current prototype works perfectly on a desktop computer, it has issues on mobile devices as the user interface is not designed for devices without pointing device or with touchscreen. We however don't think that creating a mobile version of our interface will be too difficult if the itinerary calculation is done on a remote server (approach used on the iPhone or on Android). Running the calculation on the mobile device will be more challenging as we get again a limited memory. Furthermore, or prototype requires about 15 seconds to build the graph. While it is negligible on a server (it is built only once and then kept in memory), it will be annoying for a mobile user.

Last but not least, the implementation of a multiobjective shortest path algorithm and its integration in our prototype would make difference with existing systems. As it also the most interesting part from a research point of view, we currently focus on that part.

Conclusions

We presented in this article a representation model for the multimodal shortest path problem. It allows to match closely the reality and to use existing implementation of shortest path algorithms. Benchmarks shows its excellent performances on realistic situations and scales well.

We made a small introduction to multiobjective optimization and how it would help to make simple user interfaces that match precisely user preferences without any interaction. It is therefore a great tool for mobile devices where input possibilities are limited.

Acknowledgement

We would like to thank the company SODIT (`http://www.sodit.eu`) and the region Midi-Pyrénées for their financial participation on this research topic. We also thank the OpenStreetMap community for high quality of the maps they produce.

References

[AOPS02] R.K. Ahuja, J.B. Orlin, S. Pallottino, and M.G. Scutella. Minimum Time and Minimum Cost-Path Problems in Street Networks with Periodic Traffic Lights. *Transportation Science*, 36(3):326–336, 2002.

[ASG07] Ines Alaya, Christine Solnon, and Khaled Ghedira. Ant Colony Optimization for Multi-objective Optimization Problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 450–457. IEEE Computer Society, October 2007.

[CVVL02] C.A.C. Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[FT87] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[GM] Michel. Gondran and Michel. Minoux. *Graphs, dioïds and semirings: new models and algorithms*. Springer.

[GM01] F. Guerriero and R. Musmanno. Label Correcting Methods to Solve Multicriteria Shortest Path Problems. *Journal of Optimization Theory and Applications*, 111(3):589–613, 2001.

[PS98] S. Pallottino and M.G. Scutella. *Shortest path algorithms in transportation models: classical and innovative aspects*. Kluwer Academic Publishers, 1998.

[ZW00] A. Ziliaskopoulos and W. Wardell. An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays. *European Journal of Operational Research*, 125(3):486–502, 2000.