Efficient algorithms for the 2-Way Multi Modal Shortest Path Problem

Marie-José Huguet, Dominik Kirchler, Pierre Parent, Roberto Wolfler Calvo

Efficient algorithms for the 2-Way Multi Modal Shortest Path Problem

Marie-José Huguet[1,2], Dominik Kirchler[3,4,5]
Pierre Parent[5], Roberto Wolfler Calvo[5]

[1] *CNRS, LAAS; Toulouse, France*
[2] *Université Toulouse; LAAS; France*
[3] *LIX; Ecole Polytechnique*
[4] *Mediamobile; Ivry-sur-Seine*
[5] *LIPN; Université Paris 13*

Abstract

We consider the 2-Way Multi Modal Shortest Path Problem (2WMMSPP). Its goal is to find two multi modal paths with total minimal cost, an outgoing path and a return path. The main difficulty lies in the fact that if a private car or bicycle is used during the outgoing path, it has to be picked up during the return path. The shortest return path is typically not equal to the shortest outgoing path as traffic conditions and timetables of public transportation vary throughout the day. In this paper we propose an efficient algorithm based on bi-directional search and provide experimental results on a realistic multi modal transportation network.

Keywords: 2-Way Shortest Path, time-dependency, multi modal transportation network, label constrained shortest path

[1] Email: kirchler@lix.polytechnique.fr, parent@lipn.fr
[2] Email: huguet@laas.fr, wolfler@lipn.fr

1 Introduction

Multi modal transportation networks include roads, public transportation, bicycle, etc. A *multi modal path* on such a network may consist of several transportation modes. The goal of the 2-Way Multi Modal Shortest Path problem (2WMMSPP) is to find two multi modal paths with total minimal cost, an outgoing path from, e.g., home to work in the morning, and a return path from work to home in the evening. The main difficulty lies in the fact that if a private car or bicycle is used during the outgoing path, it has to be picked up during the return path. As noticed in [2], on a multi modal transportation network the shortest return path is typically not equal to the shortest outgoing path as traffic conditions and timetables of public transportation vary throughout the day and one-way roads prevent the same path from being be taken in the opposite direction. See Figure 1 for an example.

To the best of our knowledge, the 2WMMSPP has been previously studied only in [3]. The authors of [3] adopt a brute force algorithm by calculating all paths between the start and final location and a predetermined set of possible parking places.

Our Contribution. In this paper, we propose a new bi-directional multi modal shortest path algorithm for optimally solving the 2WMMSPP. We run experiments on a realistic multi modal transportation network. Our algorithm is much more efficient than the algorithm described in [3].

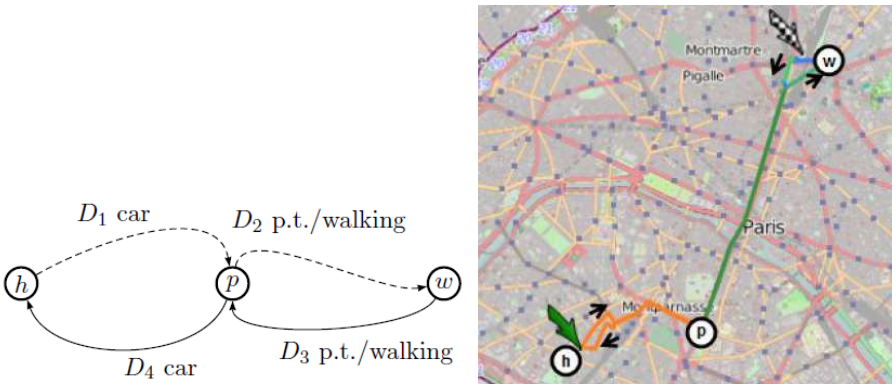

Fig. 1. Example of a 2-way path: h represents the home, p the parking place, and w the working location. For the paths between h and p only a private vehicle may be used (car or bike, orange line), and between p and w only public transportation (p.t., green line) or walking (blue line). In this example arrival time at work in the morning is set to 9am and departure time at work in the evening is set to 5pm.

2 Algorithm to solve the 2WMMSPP

Given a starting and arrival node as well as starting times and a set of transportation modes, our algorithm determines the optimal parking location and the corresponding multi modal paths. First, we discuss how to calculate multi modal shortest paths and we introduce the graph we are using. Next we discuss the basic version of our algorithm and some techniques which substantially improve its efficiency.

2.1 Multi Modal Shortest Path

Multi modal shortest paths are subject to particular constraints: passengers may want to exclude some transportation modes (e.g., the bike when it is raining) or limit the number of changes. The *regular language constrained shortest path problem* (RegLCSP) introduced by [1] deals with these kind of issues and it uses an appropriately labeled graph and a regular language to model constraints. A generalization of the well-known Dijkstra's algorithm to solve RegLCSP has been proposed (called hereafter D_{RegLC}).

Similar to [1] we use a *labeled* directed graph $G = (V, A, \Sigma)$ to model the multi modal transportation network. It consists of a set of nodes V, a set of labels Σ, and a set of labeled arcs $(i, j, l) \in A$ which are triplets in $V \times V \times \Sigma$. They represent an arc from node i to node j having label l. The labels are used to mark arcs as, e.g., foot paths, bike lanes, etc. Arc costs represent travel time. They are positive and may be *time-dependent* as we consider exact timetable information and changing traffic conditions: $c_{ijl}(\tau)$ gives the cost of an arc (i, j, l) when arriving at i at time τ.

2.2 Basic Algorithm

The *basic* version of our algorithm works as follows. We alternate the execution of four D_{RegLC} algorithms on G: algorithms D_1 and D_2 for the outgoing path, and D_3 and D_4 for the return path (see Figure 1). D_1 and D_4 calculate the shortest path from h to all other nodes by using only personal vehicle, and D_2 and D_3 from w to all other nodes by using only public transportation and foot. Algorithms D_1 and D_3 use forward search, and algorithms D_2 and D_4 use backward search. At each iteration, we choose the algorithm D_i which node x to be settled next has the lowest key among the nodes yet to be settled by the four D_{RegLC} algorithms. Node x is settled by D_i following the Dijkstra principle. Now we check if node x has been settled by all four algorithms. If this is the case a new 2-way path has been found, for which x is the parking

place. To evaluate the cost of the 2-way path it suffices to sum the costs of the 4 shortest paths to x as calculated by the 4 D_{RegLC} algorithms. If the total cost is better than the cost μ of the current best 2-way path, then μ is updated and x is memorized. The algorithm may stop as soon as the key δ of the next node to be settled is greater or equal to μ.

Time dependency. In scenarios involving time dependent transportation modes (car, public transportation) starting times of some algorithms have to be specified in order to be able to correctly evaluate time dependent arc costs. Some starting times will be known, but some others will not. For example (with reference to Figure 1), if arrival and starting time at the destination are given, then only the starting time for algorithms D_1 and D_4 are not known. Therefore if personal vehicle arcs do not depend on time we can use the basic algorithm without modifications. On the contrary if personal vehicle arcs depend on time we have to adapt the algorithm the following way. Whenever D_1 and D_4 examine time-dependent arc costs we use the minimum weight cost function $c_{ij}^{\min} = \min_{\tau \in H} c_{ij}(\tau)$, where H is the time horizon we are considering. Then, when evaluating a newly found 2-way shortest path with parking node x, we first calculate its costs as described above. We call this cost *temporary cost*. If the temporary cost is lower than μ then we *re-evaluate* the cost of the paths produced by D_1 and D_4. To do this we use two D_{RegLC} algorithms starting in x. Starting times are given by the key of x in D_2 and D_3. In this way, we obtain the real cost of the 2-way shortest path.

Complexity. Let n be the number of nodes in the graph, and m the number of arcs. Run time of the algorithm is $O(4(n \times log(n) + m))$ in scenarios which do not require re-evaluation of shortest paths. Otherwise the complexity is $O(2n \times (n \times log(n) + m))$.

2.3 Improved Stopping Condition

The algorithm may stop earlier by using a *lower bound* λ.

Proposition 2.1 *Let mc_i be the minimum total cost of a parking place settled by i of the 4 algorithms. $\lambda = \min_{i \in 1..3}(mc_i + (4 - i)\delta)$ (where δ is the cost of the next node to be settled) is a valid lower bound and the algorithm may stop when $\lambda \geq \mu$.*

Proof 1 *Any parking node which has not been settled yet by all four algorithms must have been settled either 0, 1, 2, or 3 times and its cost is at least 4δ, $mc_1 + 3 * \delta$, $mc_2 + 2 * \delta$, or $mc_3 + \delta$, respectively. Note that $4\delta \geq mc_1 + 3 * \delta$.*

Thus, as soon as $\lambda > \mu$, the optimal solution must have been found.

2.4 Further Improvements

We introduce three improvements for scenarios where re-evaluation of parking nodes is necessary.

1) Prior to starting the algorithm, we produce an upper-bound μ_0 (e.g., by choosing arbitrarily a parking node p and then calculating the four shortest paths between p, h, and w.) The better the upper-bound the sooner the algorithm will stop and the less parking nodes will have to be re-evaluated.
2) Instead of calculating the minimum weight of an arc over the entire time horizon H we may use the minimum weight of a more restricted time interval. In the example in Figure 1, arrival time t_0 and departure time t_1 at w are given. In this case we may use $c_{ij}^{min} = \min_{\tau \in [t_0 - \mu_0 + \delta, t_0]} c_{ij}(\tau)$ for D_1 and $c_{ij}^{min} = \min_{\tau \in [t_1, t_1 + \mu_0 - \delta]} c_{ij}(\tau)$ for D_4.
3) For the re-evaluation, the SDALT algorithm [6] instead of D_{RegLC} may be used. SDALT applies A^* [5] and *landmarks* [4] to speed up D_{RegLC} . It uses an estimated lower bound of the distance to the destination to guide the search more directly toward the destination.

3 Experiments and Discussion

The algorithm was implemented in C++ and compiled with GCC 4.1. Experiments are run on an AMD Opteron, clocked at 2.2 Ghz. We use a realistic multi-modal transportation network based on road and public transportation data of the French region Ile-de-France which includes the city of Paris and its suburbs. It consists of four layers: bike, foot, car, and public transportation. The four layers are connected by transfer arcs. See [7] for more information about graph models of a multi-modal network and time-dependency. Data of the public transportation network have been provided by STIF [3]. They include geographical information, as well as timetable data on bus lines, tramway's, subways and regional trains. Data for the car layer is based on road and traffic information provided by Mediamobile [4]. Circa 15% of the road arcs have a time-dependent cost function to represent changing traffic conditions throughout the day. The foot as well as the bike layer are based

[3] Syndicat des Transports IdF, www.stif.info, data for scientific use (01/12/2010)
[4] www.v-trafic.fr, www.mediamobile.fr

Scenario	Basic	Stop cond.	Imp 1	Imp 1+2	Imp 1+2+3
1) $h \leftrightarrow p$: bike, $p \leftrightarrow w$: foot	1.050	0.889	na	na	na
2) $h \leftrightarrow p$: bike, $p \leftrightarrow w$: foot, pt	2.143	1.050	na	na	na
3) $h \leftrightarrow p$: car, $p \leftrightarrow w$: foot, pt	199.5	194.6	60.4	4.7	3.7

pt: public transportation, na: not applicable, Imp: Improvement

Table 1
Average run times in seconds over 100 instances.

on road data (foot paths, bike paths, etc.) extracted from geographical data freely available from OpenStreetMap [5] . The graph consists of circa 3.7mil arcs and 1.1mil nodes. There are 270 000 possible parking places.

To evaluate run times of our algorithm, we build three realistic scenarios (see Table 1). For each scenario, we specify the allowed transportation modes for the paths between home and parking ($h \leftrightarrow p$) and between parking and work ($p \leftrightarrow w$). In all scenarios, arrival time at work for the outgoing path is 9am and departure time at work for the return path is 5pm.

In Table 1, we present the average CPU run time over 100 instances where h and w have been determined randomly. We compare the results of 5 variants of our algorithm. First we run the basic version as described in Section 2.2, to this version we added incrementally the improved stopping condition (Section 2.3) and the three improvements discussed in Section 2.4. Note that a re-evaluation of parking nodes because of time-dependent arc costs is necessary only in scenario 3, thus the three improvements apply only to that scenario.

We are able to report faster run times than those reported by the authors of [3]. Run times of their brute force algorithm are quite high even when limiting the number of possible parking nodes. They report average run times of 1min when considering 20 parking nodes and of 30min for 80 parking nodes. Note also that we work on a considerably larger graph. The average run time of our algorithm for scenarios 1 and 2 is 1sec. The improved stopping condition has an important impact on run time for these scenarios. On the other hand, it has no impact on scenario 3 as the re-evaluation of parking nodes dominates run times. Improvements 1 and 2 succeed in decreasing the number of parking nodes which have to be re-evaluated and considerably accelerate the algorithm. Improvement 3 provides a further small speed-up.

[5] See www.openstreetmap.org

4 Conclusions

We presented a new algorithm to solve the 2WMMSPP. We were able to report better run times than those reported in the literature. Future research directions include the investigation of stronger stopping conditions, of techniques to further decrease the number of parking nodes which have to be re-evaluated, and of the use of parallization.

References

[1] Christoper L. Barrett, Riko Jacob, and Madhav Marathe. Formal-Language-Constrained Path Problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.

[2] Daniel Baumann, Alexandre Torday, and Andre-Gilles Dumont. The importance of computing intermodal roundtrips in multimodal guidance systems. In *Proceedings of the 4th Swiss Transport Research Conference*, March 25-26 2004.

[3] Aurelie Bousquet, Sophie Constans, and El Faouzi Nour-Eddin. On the adaptation of a label-setting shortest path algorithm for one-way and two-way routing in multimodal urban transport networks. In *International Network Optimisation Conference*, pages 1–8, 2009.

[4] Andrew V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 156–165. SIAM, Philadelphia, 2005.

[5] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[6] Dominik Kirchler, Leo Liberti, and Roberto Wolfler Calvo. A label correcting algorithm for the shortest path problem on a multi-modal route network. In *Symposium on Experimental Algorithms (SEA)*, volume 7276 of *LNCS*, pages 236–247, 2012.

[7] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, 12(2.4), June 2008.