On the adaptation of a label-setting shortest path algorithm for one-way and two-way routing in multimodal urban transport networks

Bousquet Aurélie[1], Constans Sophie, El Faouzi Nour-Eddin

INRETS, LICIT, laboratoire d'ingénierie circulation transports
25 avenue François Mitterrand, Bron, F-69675, France

ENTPE, LICIT, laboratoire d'ingénierie circulation transports
rue Maurice Audin, Vaulx-en-Velin, F-69518, France

Université de Lyon
Lyon, F-69003, France

Abstract

Most of the current traveller information applications only consider one transport mode and one-way trips. However, multimodal information is a key element to promote the combined use of different transport modes, and, as travel times can vary over time, a two-way shortest multimodal path is not always equivalent to two one-way shortest paths computed separately. In this paper, we first present a dynamic label-setting algorithm able to compute a one-way multimodal shortest path. Then, based on this algorithm, we propose a strategy for solving the two-way problem.

Keywords: *Multimodality, shortest path, label-setting algorithm*

1 Introduction and state of the art

To efficiently guide users through multimodal transport networks, travel time is a major issue and has to be estimated door-to-door and in real time; when possible, the entire trip chain has to be considered. Ad hoc shortest paths algorithms design and implementation are thus crucial in developing multimodal information tools.

The basic model of a transport network is a directed graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. A cost function (travel time, distance...) associates to each edge $(i, j) \in E$ a value c_{ij}. To represent multimodal networks, layer graphs have been used by Barrett et al. [2] and Hobeika [8]. In these works, each layer stands for a monomodal network and the layers are bound together with transfer edges.

Classical strategies to find a shortest path in a static graph are label-setting or label-correcting algorithms. Label-setting algorithms are more adapted to the one-to-one shortest path problem, and hence to routing in transport networks, since they can be stopped when the destination node is permanently labelled. The static shortest path problem can be solved in polynomial time. A label-setting algorithm implementing the priority queue with a binary heap runs for example in $O(m \log(n))$ time in case of non-negative costs c_{ij}, where n is the number of nodes and m the number of edges in the graph.

[1]Corresponding author: aurelie.bousquet@inrets.fr

In the dynamic shortest path problem, costs associated with edges vary over time. When time is considered as a discrete variable, the previous static labelling strategies can be adapted to dynamic problems, by making a time-space expansion of the graph [10]. However, an explicit time-space representation is not an efficient strategy, since it considerably increases the size of the graph, leading to an extra calculation time. Implicit time-space representations can be used, as in the Chronological Algorithm (Chrono-SPT) proposed by Pallottino and Scutellà [10]. It should be noted that the general time-dependent shortest path problem is NP-hard. However, when costs correspond to travel times and when the graph is FIFO (First-In-First-Out), i.e. travelling along all the edges is made according to a FIFO rule, dynamic fastest path problems exhibit good properties and can be solved in polynomial time [6] [1]. In transport networks, this travel time information is calculated according to the transport mode. For private vehicles, travel times can be derived from sensor or probe vehicles data, giving a flow speed value for each link and at each time period. Determining a vehicle's travel time on an edge e using only the flow speed corresponding to the period when it enters e potentially leads to a non FIFO behaviour with no physical meaning. Sung et al. [11] propose a method to obtain a FIFO travel time function by using a combination of the flow speeds corresponding to the periods during which the vehicle stays on the edge.

When multimodal transport chains are considered, specific constraints are introduced in the shortest path problem. Indeed, in a multimodal context, the feasibility of the proposed mode sequences should be taken into account. A multimodal path should be composed of a sequence of monomodal subpaths, each of them beginning and ending in an appropriate location. It is for example impossible to start a car subpath at a place where no car is available or to end it at a place where no parking space is available. The corresponding constraints have first been defined by Battista et al. [3] and are referred to as *viability constraints*. Barrett et al. [2] show that the viable mode sequences compose a formal language, which can be represented by a finite automaton. They apply labelling algorithms to the direct product of the initial graph representing the multimodal network, and the finite automaton giving the formal language constraints. Lozano and Storchi [9] have also used an automaton in a label-correcting algorithm to solve the multimodal point-to-point shortest path problem, allowing the use of individual vehicles only from the origin node. Ziliaskopoulos and Wardell [13] have proposed efficient data structures and an algorithm to solve the multimodal shortest path problem, with proved exactness and time complexity.

The extension of the works cited above to the two-way multimodal shortest path is not straightforward, as it was underlined by Baumann et al. [4]. Indeed, in a multimodal two-way trip, private vehicles used during the first trip introduce constraints on the second one, since it is necessary to pick them up at the point where they are parked. With time-dependent costs, constraints introduced by an optimal first trip do not always lead to an optimal second trip. Figure 1 gives an example of this situation: the shortest path between O and D in the morning corresponds to a car leg from O to the park P_2 and a public transport leg from P_2 to D. However, in the evening, the return travel time via P_2 has a lot increased. In this situation, the traveller would have saved travel time on the whole trip by leaving his car in the park P_1 in the morning. If the multimodal one-way shortest path problem has been considered in some works [2] [9] [13], to our knowledge, algorithms computing two-way viable multimodal trips have never been investigated.

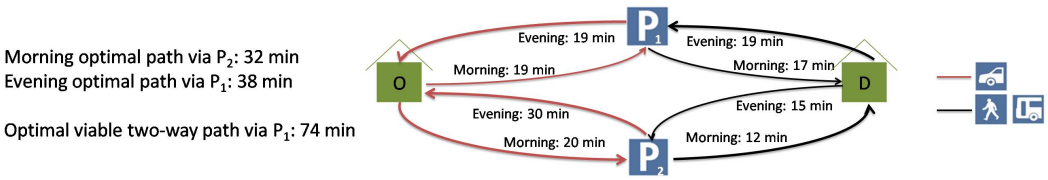

Figure 1: *Illustration of the necessity of two-ways shortest paths consideration*

In this paper, we present a label-setting algorithm able to compute a one-way multimodal shortest path and a strategy for solving the two-way problem. Our algorithm integrates viability constraints, taking into account vehicles parked anywhere in the network. Travel times are dynamic for both public

transport and individual cars and considered to be static for cycling and walking. The structure of the paper is as follows: section 2 is dedicated to the presentation of the network and travel time models, section 3 introduces the label-setting algorithm and section 4 shows applications for both one-way and two-way shortest path problems.

2 Network and travel time modelling

For the network conceptualisation, our graph structure must be able to represent all the elementary components of multimodal trips: driving, walking, cycling, searching for a parking space (on-street or in a car park), waiting at public transport stops and travelling on public transport vehicles. To do so, a structure composed of different monomodal interconnected layers is used (see Figure 2). Each layer corresponds to a transport mode (bicycle, car, walking or public transport). In the public transport layer, an edge represents a leg between two stops. In the layers associated to individual modes (walking, bicycle and car), an edge is created for each road section between two crossroads having homogeneous attributes (capacity, number of lanes, speed limit...).

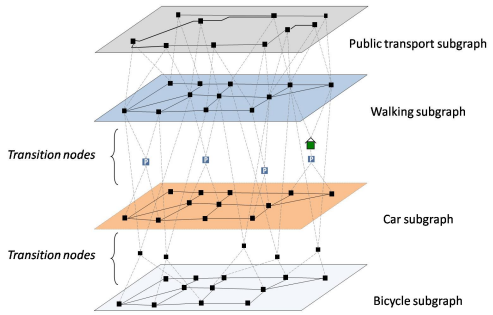

Figure 2: *Structure of the multimodal graph*

In terms of viability constraints, transport modes can be classified into two categories C_1 and C_2, depending if they use an individual vehicle (car, bicycle) or not (walking, public transport). Using a subpath of mode m, with $m \in C_1$, imposes additional viability constraints, since it requires that a vehicle is available at the beginning and a parking space can be found at the end of the subpath. Nodes representing locations where it is possible to switch from or to a mode of C_1 are added to the graph. They do not belong to any layer and are called *transition nodes*. They can stand either for car and bicycle parking facilities (on-street parking areas, car parks), for bicycle hiring points where users can take and give shared bicycles back or for private addresses where it is also sometimes possible to park a car or a bicycle. Let $T \subset N$ be the set of transition nodes. For all $i \in T$ and for all mode $m \in C_1$, two boolean attributes v_i^m and p_i^m indicate respectively if a vehicle of mode m is available in i at the beginning of the algorithm and if a parking space for a vehicle of mode m can be found in i. A monomodal subpath of mode $m \in C_1$ always begins at a node $i_o \in T$, where $v_{i_o}^m = \text{true}$ and ends at a node $i_f \in T$, where $p_{i_f}^m = \text{true}$.

Edges (i, j) have a travel time function τ_{ij} and a mode attribute $m_{ij} \in \{\text{car, bicycle, walking, public transport}\}$ associated. The travel time function can be either time-dependent or time-independent. Bicycle and walking travel times are time-independent since we consider that they do not vary with the traffic level. Car travel times are time-dependent and calculated for each edge from the flow speeds associated to the different time periods of length Δt, typically 6 minutes. A basic travel time estimation for any vehicle arriving at node i during time period k can be given by $\frac{l_{ij}}{v_{ij}(k)}$, where l_{ij} is the length of (i, j) and $v_{ij}(k)$ corresponds to the speed on (i, j) during period k. However, this estimation is too rough and leads to a non FIFO behaviour when a vehicle stays for more than one discretisation period on the

edge. Sung et al. [11] propose to calculate travel times from the speeds corresponding to all the periods during which the vehicle stays on the edge. A similar procedure is used here to build FIFO car travel times. Public transport travel times and waiting times at stops are also time-dependent, derived from timetables. As a consequence, they naturally respect the FIFO property.

Considering that the multimodal graph is built and travel times and mode attributes have been assigned to each edge, we first propose a label-setting algorithm providing a multimodal route for a one-way trip between O and D, with the following specific features:

- The solution path respects viability constraints;
- Travel times are time-independent or time-dependent, calculated from flow speeds or timetables;
- Individual vehicles located at any transition node in the network can be used in the routing process;
- It is possible to impose that a private vehicle be available in D at the end of the proposed path, since users might need it for a later trip.

Based on this algorithm, we propose a strategy for solving the two-way problem with the restriction that only one private vehicle (car or bicycle) will be used during the two trips.

3 Multimodal routing algorithm

In this section, the algorithm used in this work is introduced. We first describe a forward labelling strategy to find the one-way multimodal shortest path between O and D with the temporal constraint: leave O at time t. Note that a similar backward labelling strategy can be defined for the constraint: arrive in D at time t. O and D are transition nodes and usually correspond to address points.

A viable multimodal path can be characterised by a state depending on the modes that can be used to continue it. The six possible states are: *Car and bicycle available* (CBA), *Car available* (CA), *Bicycle available* (BA), *No vehicle available* (NA), *Using car* (UC) and *Using bicycle* (UB). Depending on the last node of the path, only particular states can be associated (see Table 1).

Last node ∈	Transition nodes	Car subgraph	Bicycle subgraph	Walking subgraph	Public transport subgraph
Possible states	*all*	UC	UB	NA	NA

Table 1: States possibly associated to a path, depending on its last node

Two kinds of transitions are possible between the states, represented by the automaton in Figure 3:

- The transitions represented by a continuous line with a label r correspond to travelling along an edge (i, j) in the multimodal graph with $m_{ij} = r$. We refer to them as *travelling transitions*;
- The transitions represented by a dashed line correspond to an update of the current state when a path arrives at a transition node where an individual vehicle can be taken or parked. We refer to them as *update transitions*.

A path from O to $i \in N$ can thus be defined by a pair (i, s_0) where s_0 is the state associated to the path. The label-setting procedure consists in scanning all the possible (i, s_0) pairs. Let $\delta^+(i)$ be the set of successor nodes for i. When a (i, s_0) pair is scanned, the nodes $j \in \delta^+(i)$ are examined and the concatenation of (i, s_0) with each edge (i, j) is tried. This concatenation corresponds to a *travelling transition* from state s_0 and with label m_{ij} in the automaton. If the transition is not possible, j is considered as unreachable from (i, s_0). Otherwise, the state s_f associated to the new path from O to j can be determined by the automaton. If j is a transition node and if a parking space or an individual vehicle is available at this node, state s_f can then be modified with a *update transition*. For example, arriving with state $s_f = UC$ at a transition node where a parking space is available implies an update to the state $s_f = CA$. Finally, the path (j, s_f) is labelled with three values:

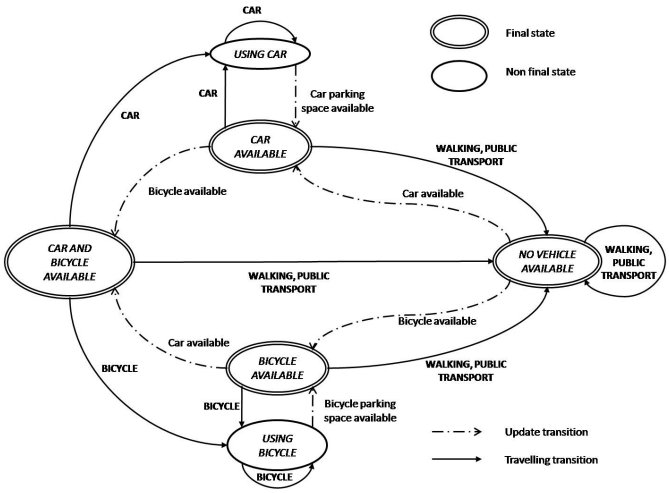

Figure 3: Automaton defining the viable mode combinations

- the potential $\pi^j_{s_f} \leftarrow \min\left(\pi^j_{s_f}, \pi^i_{s_0} + \tau_{ij}(\pi^i_{s_0})\right)$ calculated as usual in labelling algorithms,
- $pred^j_{s_f} \leftarrow i$ which gives the predecessor of node j in the path,
- $predState^j_{s_f} \leftarrow s_0$ which gives the state associated to the subpath going from O to $pred^j_{s_f}$.

The two last values are useful at the end of the algorithm to reconstitute the path from O to D.

A path ending in D cannot be labelled with states UC or UB, as each vehicle used has to be parked before arriving. If the user requests a private vehicle available (bicycle or car) in D, a modification of the automaton can be made to accept only CA or BA as final states.

Viability constraints for multimodal paths have already been defined by a finite automaton in previous works [2] [8] [9] [12]. Two additional features, which will be used by the two-way strategy, have been introduced in our automaton:

- update transitions allow to specify viability constraints for individual vehicles parked anywhere in the network and not only in O,
- a slight modification of the automaton allows to request an individual vehicle available in D at the end of the routing process.

Now, we can consider the problem of the two-way multimodal shortest path between O and D, having a vehicle of mode $m \in C_1$ (bicycle or car) available at node P_0. The vehicle must be brought back in P_0 at the end of the two-way trip. P_0 can be equal to O. To solve this problem, a list T of the N transition nodes that can be used to park the vehicle is created ($\forall i \in T$, $p^m_i = \text{true}$). Then, the proposed strategy consists in 4 steps summarised in Figure 4:

1. The shortest paths tree from O to each $i \in T$ is calculated, considering that $v^m_{P_O} = \text{true}$, with the constraint of having the vehicle available in i at the end of each path. As a consequence, the paths from O to $\{i, i \in T\}$ pass inevitably through P_0. The corresponding costs $\left\{C^i_1, i \in T \cup \{O\}\right\}$ are stored, C^O_1 being set to 0. This step requires one iteration of the one-way label-setting algorithm, which can be stopped when all the nodes in T are permanently labelled.

2. The shortest path from each $i \in T \cup \{O\}$ to D is computed, considering that the private vehicle is no longer available ($\forall i \in T$, $v_i^m =$ false). For each i, the cost C_2^i of the path from i to D is stored. This step requires N iterations (or $N+1$ if $O \notin T$) of the one-way algorithm.

3. The shortest paths tree from D to each $i \in T \cup \{O\}$ with no vehicle available ($\forall i \in T$, $v_i^m =$ false) is computed. For each i, the cost C_3^i of the path from D to i is stored. This step requires one iteration of the one-way algorithm, which is stopped when all the nodes in $T \cup \{O\}$ are permanently labelled.

4. The shortest paths from each $i \in T$ to P_0 are computed, considering for each request that a vehicle is available in i ($v_i^m =$ true) and has to be brought to P_0. Then, if $P_0 \neq O$, the shortest paths from P_0 to O with no vehicle available are computed for the different arrival times in P_0. For each i, the cost C_4^i of the path from i to O through P_0 is stored. This step requires N iterations of the one-way algorithm and N more iterations if $P_0 \neq O$.

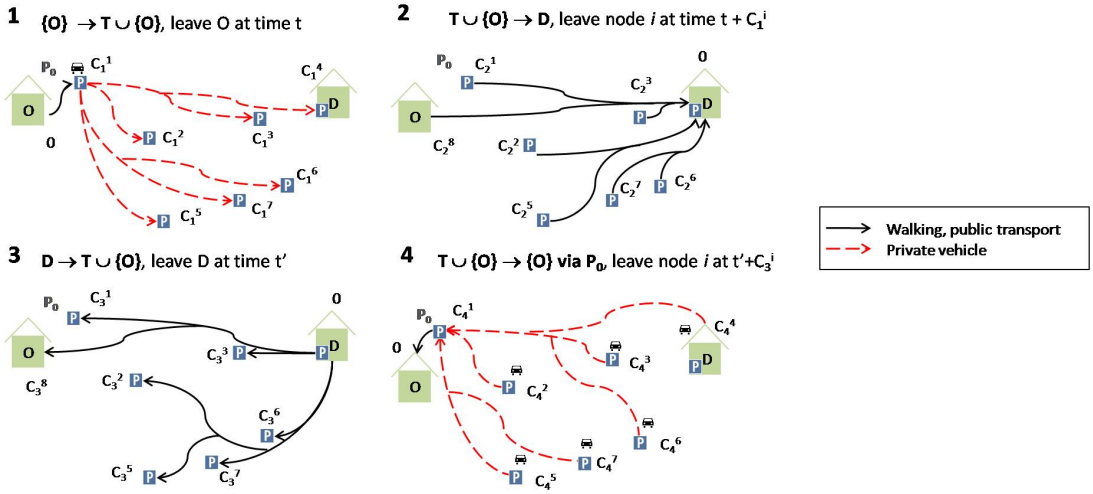

Figure 4: The 4 steps of the two-way trip computation

The node $i^* = \arg\min_{i \in T \cup \{O\}} \left(C_1^i + C_2^i + C_3^i + C_4^i\right)$ is searched. If $i^* = O$, then the optimal multimodal two-way trip corresponds to leaving the vehicle in P_0 and using only the modes *walking* and *public transport*. Otherwise, it corresponds to using the individual vehicle and parking it in i^*.

4 Application on a real-world multimodal network

The one-way and two-way multimodal shortest path algorithms have been implemented in C++ using the Boost Graph Library on a processor Intel Pentium M, 1.86 GHz, and tested on a real-world multimodal network, covering the urban area around Lyon, the second largest city in France. The size of the different layers is given by Table 2. A preprocessor builds the graph and the travel times from network data, transit timetables and traffic data stored in a database.

We propose here illustrations of multimodal paths that can be generated by our algorithm. A private car is supposed available near O in an on-street parking area P_0. First, the one-way multimodal shortest path between two addresses O and D is requested. The time constraint is: leave O at 7:30. Then the corresponding return trip, supposing that the one-way optimal trip has been realised in the morning, is requested. The time constraint is: leave D at 17:00. Finally, the two-way multimodal shortest path is requested in the same conditions. Both paths generated by the algorithm are summarised in Figure 5. We

	Car layer	Bicycle layer	Walking layer	Public transport layer	Transitions	Transfers
# nodes	$\simeq$ 48300	$\simeq$ 48300	$\simeq$ 48300	$\simeq$ 220	$\simeq$ 1500	0
# edges	$\simeq$ 118200	$\simeq$ 118200	$\simeq$ 118200	$\simeq$ 250	0	$\simeq$ 6000

Table 2: Size of the different layers of the graph

observe that choosing the optimal one-way multimodal path in the morning would lead to a suboptimal two-way multimodal route.

Optimal one-way trip		**Optimal one-way trip**	**Return trip with car parked in P_1**
Walking from O to P_0	00:02:50	Travel time = 00:23:48 Departure time = 07:30 Arrival time = 07:54	Travel time = 00:29:00 Departure time = 17:00 Arrival time =17:29
Driving to the car park P_1	00:09:38		
Parking the car	00:03:00		
Walking to D	00:08:20		
Total travel time for the two-way trip : 00:52:48			

Two-way optimisation : total travel time for the two-way trip = 00:50:50			
First trip : Travel time = 00:25:10 Departure time = 07:30 – Arrival time = 07:55		Second trip : Travel time = 00:25:40 Departure time = 17:00 – Arrival time = 17:26	
Walking from O to P_0	00:02:50	Walking from D to a bicycle hiring point	00:01:00
Driving to the car park P_2	00:06:10	Cycling to another hiring point and giving the bicycle back	00:02:10
Parking the car	00:03:00	Walking to an underground station	00:01:00
Walking to an underground station	00:02:00	Waiting	00:01:00
Waiting	00:01:00	Travelling on the underground	00:06:00
Travelling on the underground	00:06:00	Walking to car park P_2	00:02:00
Walking to a bicycle self-service hiring point	00:01:00	Driving to P_0	00:06:40
Cycling to another hiring point and giving the bicycle back	00:02:10	Parking the car at P_0 (on street)	00:04:00
Walking to D	00:01:00	Walking to O	00:02:50

Figure 5: Multimodal shortest path obtained for the one-way and the two-way requests

A one-way shortest path request takes between 1 second and 1 minute depending on the distance between O and D. For the two-way request introduced in Figure 5, the computation time increases slowly with N, when N is below 50. Beyond 50 transition nodes, the algorithm becomes time consuming, which makes it unappealing for users' information applications (see Table 3).

5 Conclusion

We have proposed a label-setting strategy to find a solution to the two-way multimodal shortest path problem. The algorithm requires at most $(3N + 3)$ iterations of the one-way algorithm, which is an adaptation of Dijkstra's algorithm. This work, which constitutes a first attempt to solve the two-way multimodal shortest path problem, illustrates the necessity of developing algorithms able to deal with

N (number of transition nodes)	10	20	30	40	50	60	70	80
Computation time (sec)	52	66	109	182	355	826	1142	2221

Table 3: Computation time for the two-way request of Figure 5 depending on the number of transition nodes considered

entire trip chains. Our future work focuses on other optimisation approaches to prove the exactness of the algorithm and to improve its computational efficiency. More condensed representations of the different monomodal networks and of the intermodal transfers could for example be envisaged, as it is done in [13].

Acknowledgement: This research is partially supported by the Rhône-Alpes Region. The authors would like to thank SYTRAL and Grand Lyon for providing the real-world data used in this work.

References

[1] Ahuja R. K., Orlin J. B., Pallottino S. and Scutellà M. G., *Dynamic shortest paths minimizing travel times and costs*, Networks, 41(4), pp. 197-205, 2003.

[2] Barrett C., Bisset K., Jacob R., Konjevod G. and Marathe M., Classical and contemporary shortest path problems in road networks: implementation and experimental analysis of the TRANSIMS router, In *Proceedings of ESA 2002, 10th Annual European Symposium*, 17-21 Sept. 2002, Springer-Verlag.

[3] Battista M.G., Lucertini M. and Simeone B., Path composition and multiple choice in a bimodal transportation network, In *Proceedings of the 7th WCTR, Sydney*, 1995.

[4] Baumann D., Torday A., Dumont A.-G., The importance of computing intermodal roundtrips in multimodal guidance systems, *Swiss Transport Research Conference*, 2004.

[5] Bellman R., *On a Routing Problem*, Quarterly of Applied Mathematics, 16, pp. 87-90, 1959.

[6] Chabini I., *Discrete dynamic shortest path problems in transportation applications: complexity and algorithms with optimal run time*, Transportation Research Records, 1645, pp.170-175, 1998.

[7] Dijkstra E. W., *A note on two problems in connexion with graphs*, Numerische Mathematik, 1959.

[8] Hobeika A., *TRANSIMS Fundamentals: chapter 5, route planner*, Tech Report, Virginia Polytechnic University, 2005.

[9] Lozano A., Storchi G., *Shortest viable path algorithm in multimodal networks*, Transportation Research, part A, 35, pp. 225-241, 2001.

[10] Pallottino S., Scutellà M. G., *Shortest Path Algorithms in Transportation models: classical and innovative aspects*, Tech Report, University of Pisa, 1997.

[11] Sung K., Bell M.G.H., Seong M., Park S., *Shortest paths in a network with time-dependent flow speeds*, European Journal of Operational Research, 121, pp. 32-29, 2000.

[12] Sherali H. D., Hobeika A. G., Kangwalklai S., *Time-Dependent, Label-Constrained Shortest Path Problems with Applications*, Transportation Science, INFORMS, 37, pp. 278-293, 2003.

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