A bidirectional/multi-queue algorithm for the bi-objective multimodal viable shortest path problem

Fallou Gueye, Christian Artigues, Marie-José Huguet, Frédéric Schettini, Laurent Dezou

A bidirectional/multi-queue algorithm for the bi-objective multimodal viable shortest path problem

Fallou Gueye[1,2,3],Christian Artigues[1,2], Marie-José Huguet[1,2], F. Schettini[3], L. Dezou[3]

[1]CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2]Université de Toulouse ; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
[3]MobiGIS; ZAC Proxima, rue de Lannoux, 31310 Grenade Cedex France
fgueye@mobigis.fr, artigues@laas.fr, huguet@laas.fr

Abstract

Taking into account the multimodality of urban transportation networks for computing the itinerary of an individual passenger introduces a number of additional constraints such as restriction and/or preferences in using some modes. Such constraints are gathered under the concept of viable path modeled by a deterministic finite state automaton. In this paper we propose several polynomial algorithms to tackle a bi-objective problem where the goal is to find all the non-dominated viable paths under the two objectives "travel time" and "number of modal transfers". These algorithms are a variant of a topological label-setting algorithm provided by Lozano and Storchi [5], a new multi-queue algorithm, as well as its bidirectional variant. The different algorithms are compared on a real network. The results show that the proposed algorithms are efficient and well suited for practical use. The proposed bidirectional algorithm outperforms other algorithms.
keywords: bi-objective viable shortest path, multi-modal transportation, multi-queue label setting algorithms, deterministic and non-deterministic finite state automaton, bidirectional search

1 Introduction and problem definition

Taking into account the multimodality of urban transportation networks for individual passenger's itinerary computation introduces a number of additional constraints such as restriction and/or preferences in using some modes. Such constraints are gathered under the concept of a viable path. In this paper we propose algorithms to tackle a bi-objective problem where the goal is to find all the non-dominated viable paths under the two objectives, namely, "travel time" and "number of modal transfers". The problem has been previously considered by Lozano and Storchi [5] who proposed an extension of the topological algorithm proposed by Pallottino and Scutellà [7]. Section 2 describes precisely the considered bi-objective multimodal viable shortest path problem. In section 3, we briefly review the state-of-the-art models and methods to deal with the multimodal characteristics. In Section 4, we propose several algorithms to solve the problem: a topological label-setting algorithm based on the Lozano and Storchi algorithm which incrementally compute solutions from 0 to a maximum number of transfers, a multi-queue algorithm

which computes firstly the shortest path with the highest number of transfers and then the other paths with less transfers, as well as a bidirectional variant of the multi-queue algorithm. In Section 6, the different algorithms are compared on a real network.

2 The bi-objective multimodal viable shortest path problem (BI-MM-V-SPP)

2.1 Problem definition

We consider a layered network $G(V, E)$ such that each layer corresponds to a mode $m \in M$. For our experimental evaluation on a real network, we consider the case where $M = \{wa, bu, pr, me\}$ (walking, bus, private car, metro). A mode $m_i \in M$ is defined for each node $i \in V$ and a travel time d_{ij} is associated to each arc $(i, j) \in E$. An arc (i, j) such that $m_i \neq m_j$ is called a transfer arc. A viable (or feasible) path is a path in G from an origin node O to a destination node D verifying in addition some multimodal restrictions. In terms of multimodal characteristics, a path in G is described by a sequence (or string) of modes, e.g. pr, wa, bu, wa. Among all mode strings of modes, only a subset of strings are acceptable according to a passenger's preferences. A generic way of representing feasibility of the mode sequences is through the use of a finite state automaton (FSA) that validates or not the mode string. A path issuing a valid mode string is named a viable path (as in [5]), i.e. a feasible path satisfying the multimodal restrictions.

The input finite state automaton is given by a 5-uple $A = (S, M, \delta, s_0, F)$ where $S = \{1, \ldots, |S|\}$ is the set of states, s_0 is the initial state, F is the set of final states and $\delta : M \times M \times S \rightarrow S$ is the transition function such that $\delta(m, m', s)$ gives the state obtained when traversing from state s an arc (i, j) with $m_i = m$ and $m_j = m'$. A path is viable if it starts with O (in state s_0) and reaches D in a state $s \in F$.

We consider both the "minimum time" and "minimum number of transfers" objectives. We first recall definitions on multiobjective optimization [4] applied to our problem. Let $time(p)$ denote the travel time along a path p. Let $ntr(p)$ denote the number of transfers along p. An efficient (or Pareto-optimal) solution is a feasible O-D path p such that there is no other path p' verifying either $time(p') \leq time(p)$ and $ntr(p') < nbtr(p)$, or $time(p') < time(p)$ and $ntr(p') \leq nbtr(p)$. In the objective space, a non-dominated point is a pair (t, k) such that there exists an efficient path p verifying $time(p) = t$ and $ntr(p) = k$.

Considering the bi-objective "minimum time" and "minimum number of transfers" O-D viable path problem, the goal is to find all non-dominated points, and, for each of them, a single efficient path.

2.2 Example of modelling path viability through finite state automatons

We consider at first the deterministic finite state automaton with $|S| = 5$ represented in the left part of Figure 1 and given by Lozano and Storchi [5]. Transition arcs between states are labeled by a mode $m \in M$ where $M = \{wa, bu, pr, me\} \cup \{O\}$ (walking, bus, private car, metro) and mode at the origin ($m_O = O$). This automaton represents constraints on metro and private modes. The considered intineraries are assumed to be from home to another place. Hence, the private mode can be taken only from O and, once left, cannot be taken again. For the metro, one assume that it can be taken at any time but, once left, cannot be taken again. A transition from state s to state s' labeled by $m \in M$ describes

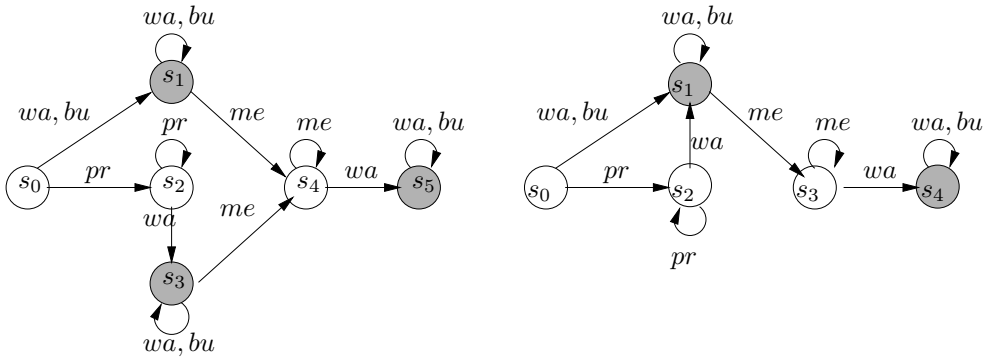

Figure 1: Original and reduced deterministic finite state automaton

the transition function of a traversed arc (i, j) in such a way that $s' = \delta(m_i, m_j, s)$ with $m_j = m$.

If a mode m does not appear as a possible transition of a given state s, any transition towards this mode is forbidden. The state at origin is s_0. State s_1 means that private car was not taken at O and so mode pr is forbidden for the remaining of the travel, while metro has not been taken yet. State s_2 means that private car was taken at O and has not been left yet. State s_3 means that private cannot be taken anymore since it has already been taken and left while metro mode has not been taken yet. In state s_4 metro has been taken but not left. In state s_5, metro has been left. We consider the acceptable final states are reduced to $F = \{s_1, s_3, s_5\}$ (displayed in grey in Figure 1). Indeed, state s_4 models the presence of the user in the metro, so she/he must leave the metro to reach her/his destination. State s_2 means the private car is currently being used and must be left in a parking area to reach the destination.

The worst case time complexity of the algorithms proposed in the following sections is a function of the number of states. Hence, avoiding redundant states is an important issue. Given the specifications of path viability, the automaton proposed by Lozano and Storchi [5] displayed in the left part of Figure 1 can be reduced as we can prove that states $s1$ and $s3$ are equivalent. This is done through state-based dominance rules explained in Section 4.2). Hence the automaton displayed in the right part of Figure 1 describes the same viable path as the one proposed in [5] with a smaller number of states ($|S| = 4$ and $F = \{s_1, s_4\}$). So this automaton will be the part of the data of our case-study.

3 Literature review

A basic version of the considered problem is the case where there is no multimodal restrictions (all mode strings are valid and all paths are viable). Pallottino and Scutellà [7] observe that the possible values for the "minimum number of transfers" is discrete and finite from 0 to $k_{\max}$ if $k_{\max}$ denotes an upper-bound on the number of transfers. They propose a so-called topological algorithm computing the minimum time paths in increasing order of the number of transfers k. The overall time complexity is $O(|E|nk_{\max})$. Note that $k_{\max}$ is bounded from above by n.

Besides the number of transfers objective, multimodality may induces constraints or user preferences in successive modes appearing in a path. A first class of approaches

models such path viability constraints by means of mode-dependent travel times and mode-dependent switching delays, in a time-dependent context [10].

Another class of approaches consider the network is layered such that arcs and node a each layer correspond to a particular mode while transfer arcs link the nodes of different layers to model mode switch. The label-constrained shortest-path problem [1, 2, 8, 9] allows to tackle in a general way this characteristics. Without considering explicitely the modes, this extension of the shortest path problem considers a graph $G(V, E)$, an alphabet Σ and a regular language L. Each arc (i, j) is valuated by a travel time d_{ij} and an element of the alphabet $a_{ij} \in \Sigma$. The problem consists of finding a shortest path p from O to D such that the concatenated labels along the path form a word of L. The regular language can be used to model path viability in terms of mode. Barett *et al* [1] provide a simple example modeling the viable paths consisting in walking from O, then taking a bus with no transfer and, lastly, walking to D by the regular expression w^*bw^*, where $w, b \in \Sigma$ represent walking and bus arcs respectively.

A regular language can be represented by a non-deterministic finite state automaton (FSA) $A = (S, \Sigma, \delta, s_0, F)$ with a set of states S an initial state s_0 and a set of final states F, and a transition function $\delta : \Sigma \times S \rightarrow 2^S$. Barett *et al.* [2] prove the problem is polynomial using the product graph $G \times A$ with nodes (i, s) for each $i \in V$ and $s \in S$ such that there is an arc from (i, s) to (j, s') if there is an arc (i, j) in E and a transition such that $s' \in \delta(a_{ij}, s)$. Under this definition the problem resorts to finding a shortest path between (O, s_0) and (D, s) with $s \in F$. In [1, 8, 9], practical implementation issues of this method are discussed. Barett *et al.* [1] propose $A*$ and bidirectional accelerations. Considering deterministic FSA as input, Sherali *et al.* [9] extend the problem to time-dependence and propose a strongly polynomial algorithm for FIFO graphs. Sherali *et al.* [8] further extend the problem to approach-dependent travel times and propose a label-setting algorithm which consistently outperforms a label correcting algorithm designed for the same problem. Independently of this class of approaches that consider only the minimum time criterion, Lozano and Storchi [5] present an extension of the bi-objective multimodal shortest path problem (see [7] and Section 4.3) to path viability modeled by a FSA and a mode-layered graph. The proposed algorithm is a direct extension of the Palottino and Scutellà [7] topological method. We describe this algorithm in Section 2. Last, Bielli *et al* [3] consider a simplified version of the FSA model but include time-dependent arcs and time penalties for turning movements. The objective is to compute the $K-$ shortest paths under an upper bound of the maximum allowed number of transfers. The method can also be defined as an extension of the topological Pallottino and Scutellà algorithm, with labels on arcs. Experimental validations are limited to small networks. The largest one, presented in [3], involves 1000 nodes and 2830 arcs and the K-shortest path algorithm runs in 6.5s on a Pentium II with 64 MB RAM. To our knowledge, no realistic computational experiments were carried out for the bi-objective multimodal shortest path problem. One of main purposes of this paper is to carry out such experiments.

4 Algorithms for the BI-MM-V-SPP

4.1 Label setting algorithms

The proposed algorithms use labels to represent paths. Let (i, s, k) denote a label representing a path from the source to node i in state s and using k transfers. Each label has two attributes: t_{is}^k which denotes the arrival time on i and p_{is}^k which denotes the predeces-

sor label of (i, s, k) on the path. When $(j, s', k') = p_{is}^k$ it means that (a) arc (j, i) is used on the path and that (b) the state of the path on j is s' with $s' = \delta(m_j, m_i, s)$ and that (c) there is k' used transfers with $k' = k$ if $m_i = m_j$ and $k = k' + 1$ if $m_i \neq m_j$. Note that no algorithm needs to store more than one label (i, s, k) for fixed i,s and k. Consequently, the considered bi-objective problem is polynomial and all the proposed algorithms are of polynomial time complexity. We opt for the label setting principle which is a simple extension of Dijkstra algorithm incorporating the multimodal restrictions and the number of transfers computations, described hereafter.

Initially, a label $(O, s_0, 0)$ is generated with $t_{OS_0}^0 = 0$ and $p_{OS_0}^0 = (O, s_0, 0)$. The label is stored in a convenient data structure Q (see the different algorithms in the subsequent sections). The label setting process is then applied until Q becomes empty. At each iteration, the label (i, s, k) with minimum t_{is}^k is removed from Q as t_{is}^k is the shortest time from O to i in state s with number of transfers k. Then, the direct successors of node i in G are scanned. For each successor j, we first check if taking arc (i, j) is feasible according to multimodal restrictions which is true if $s' = \delta(m_i, m_j, s) \neq \emptyset$. If label extension through j is viable, we set the number of transfers k' at j to k if $m_i = m_j$ or to $k + 1$ otherwise. We obtain a label (j, s', k'). we set $t_{js'}^k := t_{is}^k + d_{ij}$ and $p_{js'}^{k'} := (i, s, k)$ if the label was never visited or if $t_{js'}^k < t_{is}^k + d_{ij}$. The the label is inserted in Q if some dominance rules do not apply. Otherwise the label is discarded.

4.2 Dominance rules and state reduction

We can state dominance rules allowing to discard labels. The basic dominance rule is linked to the bi-objective optimization.

Proposition 4.1 (Basic dominance rule) *Consider two disctinct labels (i, s, k) and (i, s, k'). If $k \leq k'$ and $t_{is}^k \leq t_{is}^{k'}$, (i, s, k') can be discarded.*

Obviously, under the described conditions, any O-D path issued from (i, s, k) is non-dominated by any O-D path issued from (i, s, k').
The state-based dominance rule strengthens the basic dominance rule considering label extension possibilities in terms of multimodal restrictions. We consider a binary relation $\preceq$ on the states such that $s \preceq s'$ means that s yields more extension possibilities than s'. More precisely $s \preceq s'$ if for any mode pair $(m, m') \in M$ one of the following conditions holds

$$\begin{cases} \delta(m, m', s') = \emptyset \\ \delta(m, m', s') = \delta(m, m', s) \\ \delta(m, m', s) = s \wedge \delta(m, m', s') = s' \end{cases}$$

Proposition 4.2 (State-based dominance rule) *Consider two disctinct labels (i, s, k) and (i, s', k'). If $k \leq k'$, $t_{is}^k \leq t_{is}^{k'}$, $s \preceq s'$ (i, s', k') can be discarded.*

From the state-based dominance rule, we also derive the following state merging conditions that were used to reduce the automaton initially proposed by Lozano and Storchi [5], as explained in Section 2.

Proposition 4.3 *If $s \preceq s'$ and $s' \preceq s$, s and s' can be merged into a single state*

We remark in Figure 1 that states $s1$ and $s3$ verify the above-described condition.

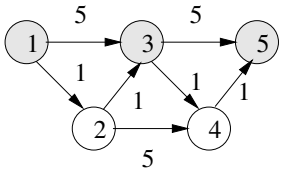

Figure 2: A bi-objective multimodal shortest path problem

4.3 Topological label-setting (TLS) algorithm

The topological Pallottino and Scutellà [7] algorithm was extended by Lozano and Storchi algorithm [5] to path viability. We describe below the algorithm TLS.

Under the topological principle, the data structure Q storing labels is made of two priority queues Q^{now} and Q^{next}. Labels are generated according to the increasing number of transfers. Initially, Q^{now} contains label $(O, s_0, 0)$ while Q^{next} is empty. At a typical iteration, the minimum time label (i, s, k) is taken from Q^{now}. Each non-dominated extended label (j, s', k'), j being a direct successor of i, is queued into Q^{now} if $k = k'$ and into Q^{next} if $k' = k + 1$. As soon as the destination D is dequeued from Q_{now} or Q_{now} becomes empty, Q_{now} is set to Q_{next} and Q_{next} is emptied. The algorithm stops when Q_{next} is already empty meaning that no non-dominated labels with $k + 1$ transfers could be found, or when a maximum number of transfers is reached.

The basic dominance rule to decide whether (j, s', k') is kept or discarded can be performed in $O(1)$: for a given pair (i, s), we have only to keep track of the shortest time found so far to reach (i, s) with $k' \leq k$ tranfers, denoted $lastlabel_{is}$, and the label is dominated if $t^{k'}_{js'} \geq lastlabel_s$, as the previously encountered label cannot have more transfers.

The algorithm pseudo code is given in Appendix (Algorithm 1).

We now establish the complexity of our implementation of TLS using binary heaps for Q_{now} and Q_{next}. Let k_{max} denotes the maximum allowed number of transfers. Note k_{max} is bounded from above by n. For a given number of transfers k, at most $n|S|$ labels (i, s, k) are selected as minimum time labels in Q_{now}. For each of them, there are two operations : (a) deletion from Q_{now} and (b) successor scan and insertion in Q_{now} or Q_{next}. Deletion from the binary heap takes $\log n|S|$ time. Successor scan with the basic dominance rule (in $O(1)$) and possible insertion (in $O(\log n|S|)$ takes $|FS_i| \log n|S|$ time where FS_i is the set of direct successors of i. Hence there are a maximum number of $n_a = n|S| \log n|S|$ (a) operations and a maximum number of $n_b = |S||E| \log n|S|$ with $n_b >> n_a$. It follows that the worst-case time complexity of TLS with the basic dominance rule and binary heap implementation is $O(k_{max}|S||E| \log n|S|)$. Running the state-based dominance rule takes in addition —S— operations for each successor so we obtain in this case a worst-case complexity of $O(k_{max}|S||E|(\log n|S| + |S|))$.

To illustrate the TLS algorithm behavior, consider the bi-objective multimodal shortest path problem from node 1 to node 5, represented in Figure 2 (there are no multimodal restrictions). There are 2 modes and 5 nodes and the shortest path is obtained for the maximum number of transfers $k = 4$.

We present below the by step-by-step execution of Algorithm TLS.

$k = 0$

$i = 1$, $Q^{now} = \{3\}$, $t^0_3 = 5$, $Q^{next} = \{2\}$, $t^1_2 = 1$

$i = 3$, $lastlabel_3 = 5$, $Q^{now} = \{5\}$, $t_5^0 = 10$ $Q^{next} = \{2, 4\}$, $t_4^1 = 6$
$i = 5$, $lastlabel_5 = 10$ (shortest path with 0 transfer)
$k = 1$, $Q^{now} = \{2, 4\}$, $t_2^1 = 1$, $t_2^1 = 6$, $Q^{next} = \emptyset$
$i = 2$, $lastlabel_2 = 1$, $Q^{now} = \{4\}$, $Q^{next} = \{3\}$, $t_3^2 = 2$
$i = 4$, $lastlabel_4 = 6$, $Q^{now} = \emptyset$, $Q^{next} = \{3, 5\}$, $t_5^2 = 7$
$k = 2$, $Q^{now} = \{3, 5\}$, $t_3^2 = 2$, $t_5^2 = 7$
$i = 3$, $lastlabel_3 = 2$, $Q^{now} = \{5\}$, $Q^{next} = \{4\}$, $t_4^3 = 3$
$i = 5$, $lastlabel_5 = 7$ (shortest path with 2 transfers)
$k = 3$, $Q^{now} = \{4\}$, $t_4^3 = 3$
$i = 4$, $lastlabel_4 = 3$, $Q^{now} = \emptyset$, $Q^{next} = \{5\}$, $t_5^4 = 4$
$k = 4$, $Q^{now} = \{5\}$, $t_5^4 = 4$
$i = 5$ $lastlabel_5 = 4$ (shortest path with 4 transfers)
$k = 5$, $Q^{now} = \emptyset$

4.4 Multi-queue label-setting (MQLS) algorithm

The topological algorithm TLS computes the non-dominated shortest paths in increasing order of the number of transfers. We propose an alternative multi-queue algorithm that computes the shortest paths in increasing order of the time criterion values. Instead of considering Q^{now} and Q^{next}, we build incrementally a list $\mathcal{Q} = \{Q_0, Q_1, \ldots\}$ of priority queues (implemented as binary heaps) such that $Q_k \in \mathcal{Q}$ contains labels representing paths with k transfers. More precisely, Q_0 is initialized with label $(O, s_0, 0)$, all other Q_k being empty. The upper bound of the number of transfers K is set to $k_{\max}$. At each iteration the label (i, s, k) with minimum travel time is taken among all non-empty priority queues. If a destination label (D, s, k) is dequeued, priority queues $Q_{k'}$ with $k' > k$ are discarded and K is set to $k - 1$, as the shortest path with k transfers to D is found. Otherwise, non-dominated labels (j, s', k') such that $k' \leq K$ issued from (i, s, k) are inserted in the corresponding priority queue $Q_{k'}$. Algorithm stops when the shortest path with 0 transfer is found or all queues are emptied. The algorithm pseudo code is given in Appendix (Algorithm 2).

Now we determine the algorithm complexity, using binary heaps for each $Q_k \in \mathcal{Q}$. At most $k_{\max} n|S|$ labels are dequeued (marked), and for each of them a minimum search operation in the $k_{\max}$ queues (a) is followed by a deletion operation in $O(\log n|S|)$ (b) and a successor scan operation (c). For each scanned successor, a dominance check (c.1) is possibly followed by an insertion operation in the appropriate queue (c.2). The basic dominance check can be made here in at most $k_{\max}$ operations as all labels (j, s', k'') with $k'' \leq k'$ must be checked. So the successor scan operation (c) as an $O(|FS_i|(k_{\max} + \log(n|S|))$ worst-case time complexity. Taking account of (a) and (b) operations we obtain a worst-case complexity of

$$O(k_{\max}|S|\,(nk_{\max} + n\log n|S| + |E|k_{\max} + |E|\log n|S|) = O(k_{\max}|S||E|\,(k_{\max} + \log n|S|)\,.$$

The time complexity is increased compared to the topological algorithm by a $k_{\max}$ factor. When the state-based dominance rule is applied, the number of operations for dominance check is multiplied by $|S|$, so we obtain a worsdt case time complexity of $O(k_{\max}|S||E|\,(k_{\max}|S| + \log n|S|)\,.$

Considering the example of figure 2, we show hereafter the step-by-step execution of this algorithm for which the shortest path is found at earlier iterations compared to the TLS algorithm.

Algorithm 2:
$Q_0 = \{1\}$
$i = 1, k = 0$, $Q_0 = \{3\}$, $t_3^0 = 5$, $Q_1 = \{2\}$, $t_2^1 = 1$
$i = 2, k = 1$, $Q_1 = \{4\}$, $t_4^1 = 6$, $Q_2 = \{3\}$, $t_3^2 = 2$
$i = 3, k = 2$, $Q_2 = \{5\}$, $t_5^2 = 7$, $Q_3 = \{4\}$, $t_4^3 = 3$
$i = 4, k = 3$, $Q_3 = \emptyset$, $Q_4 = \{5\}$, $t_5^4 = 4$
$i = 5, k = 4$, $Q_4 = \emptyset$ (shortest path with 4 transfers)
$i = 3, k = 0$ $Q_0 = \{5\}$, $t_5^0 = 10$
$i = 4, k = 1$, $Q_1 = \emptyset$
$i = 5, k = 2$, $Q_2 = \emptyset$ (shortest path with 2 tranfers)
$i = 5, k = 0$, $Q_0 = \emptyset$ (shortest path with 0 tranfers)

We show the equivalence of TLS and MQLS in the sense they both have the nice feature described by the following property. As in the standard Dijkstra algorithm, a label is "marked" as soon as it is dequeued from Q.

Proposition 4.4 *The set of labels (i, s, k) marked by TLS or MQLS for a given (i, s) maps the set of all non-dominated points for the biobjective $O - i$ viable path problem with s as final state.*

In particular, setting $i = D$ and $s \in F$, we see that TLS and MQLS generates one and only one path for each non-dominated point.

5 Bidirectional Algorithm

We propose an adaptation of MQLS in a bidirectional way taking advantage of the multi-queue characteristics and the FSA. The proposed bidirectional algorithm (FB-MQLS) maintains, in a similar way as in MQLS algorithm, two priority queue lists $\mathcal{FQ}$ for the forward search and $\mathcal{BQ}$ for the backward search such that FQ_k contains forward labels $ft_{i,s}^k$ representing paths reaching i in state s with k transfers and BQ_k contains backward labels $bt_{i,s}^k$ representing paths originating from i with k transfers in state s. There are two main issues in designing a bidirectional algorithm for the considered multimodal problem.

The first issue is linked to modeling backward path viability. We exhibit below two different possibilities.

The first possibility is simply to reverse the arcs in the finite state automaton. Generally, the obtained state automaton is non-deterministic (see left part of Fig.3). The start state (at destination) is S_5. Final states are s_1 (departure by walk or bus) and s_2 (departure by private car). Transition function $\delta(m_i, m_j, s)$ gives a set of possible states. For example $\delta(D, wa, s_5) = \{s_1, s_4\}$ (where $D = m_D$). This means what when arriving by walk at the destination, it could be or not that the metro was taken (state s_4) or note (state s_1). In practice, each time a label extension uses an arc that yields several possible successor states (in the backward path), all the corresponding labels are generated. Note that such an indetermism may yield pairs (i,s) that may never reach the origin, inducing useless computations.

The second possibility is to use a deterministic finite state automaton for the backward search. This is always possible as there exist algorithms that transform a non-deterministic finite state automaton equivalent to any deterministic one. An issue then is to generate the deterministic automaton with a minimal number of states. We display in right part

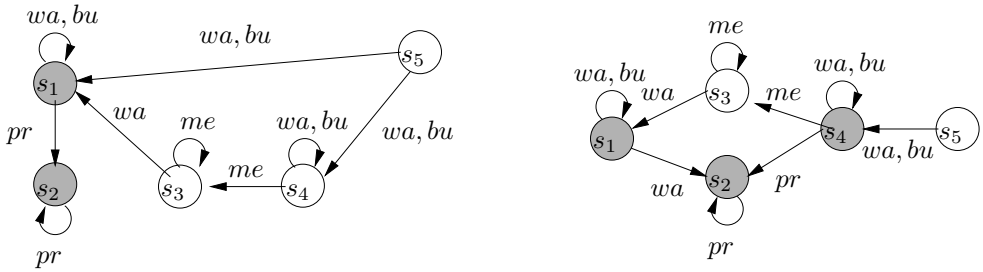

Figure 3: Non-deterministic and deterministic backward finite state automaton

of Fig.3 a possible deterministic FSA for backward path involving the same number of states.

The second issue is linked to connection consequences between a forward label and a backward label in terms of number of transfers. In this case the interest of the multi-queue implementation appears. Indeed, when a connection is made between a label $ft^k_{i,s}$ and a label $bt^q_{i,s}$, if condition

$$ft^k_{i,s} + bt^q_{i,s} \leq \min_{(i',s',k') \in \mathcal{FQ}} ft^{k'}_{i',s'} + \min_{(i',s',k') \in \mathcal{BQ}} bt^{k'}_{i',s'}$$

holds, all priority queues $FQ_{k'}$ and $BQ_{k'}$ with $k' \geq k + q$ can be discarded.

$\mathcal{FQ}$ is initalized to a single priority queue F_0 with a single label $(O, s_0, 0)$ and $\mathcal{BQ}$ is initialized to a single priority queue B_0 with a labels $(D, s_D, 0)$. The upper bound of the number of transfers K is set to $k_{\max}$. The main loop computes the minimum time forward label (i^f, s^f, k^f) and the minimum time backward label (i^b, s^b, k^b). The search proceeds from the minimum time label among them (i, s, k). The minimum time label (i, s, k) is removed from its priority queue (FQ_k or BQ_k).

Then label extension is performed in a similar way as in Algorithm MQLS, except that, for each new label (j, s', k'), a connection with the opposite direction search is searched by scanning all labels (j, s', k'') with $k' + k'' \leq K$ which possibly yield an $O - D$ path of less than K transfers. If such a connection is established, the path time ($ft^{k'}_{j,s'} + bt^{k''}_{j,s'}$ or $bt^{k'}_{j,s'} + ft^{k''}_{j,s'}$) is compared against the best $O-D$ path already found with $k'+k''$ transfers whose time is stored in an array $L^{k'+k''}$ to possibly update it.

In this case, an optimality test can be performed by comparing the minimum time L^{k^*} obtained among the extensions, with a lower bound given by the sum of the minimum forward and backward label times ($ft^{k^f}_{i^f,s^f} + bt^{k^b}_{i^b,s^b}$). If the test is positive, L^{k^*} is the best path time for k^* transfers and priority queues $FQ_{\tilde{k}}$ and $BQ_{\tilde{k}}$ with $\tilde{k} \geq k^*$ can be discarded.

Algorithm FB-MQLS pseudo code is given in the Appendix (Algorithm 3).

6 Computational Experiments

6.1 Network and data set

The aim of these experiments is to compare TLS, MQLS and FB-MQLS both on deterministic (FB-MQLS-D) and non deterministic (FB-MQLS-ND) FSA and to evaluate the efficiency of the dominance rules.

The experimental comparisons were carried out on a real-world multimodal network covering a part of the urban area of Toulouse (France) with 63048 nodes and 159368 arcs. Considered modes are bus, metro, walking and private vehicle. Timetables for buses and metro are approximate by a average travel time for each corresponding arc in the network. In this network, private vehicle can only stop at parking nodes (which is consistent with the FSA). Table 1 details the different layers of the graph in terms of modes, nodes and arcs.

Modes	Nodes	Arcs
Bus	6170	6646
Metro	75	72
Street	56774	146280
Transfer	-	6370
Parking	29	-
Total	63 048	159 368

Table 1: *Network data*

Experiments concern 100 origin-destination pairs whuch have been randomly generated relatively far apart. $k_{\max}$ is set to 10 transfers. About global results (the same obtained by all algorithms), average minimum (maximum) travel times are of about 178 (818) min. Solutions have from 0 to 8 modal transfers, with an average of 2.5 transfers. The number of non-dominated solutions vary from 5 to 7 with 5.68 non-dominated solutions per itinerary on average.

All algorithms have been implemented in C++ on an 2.47 GHz Intel Xeon processor W3520 with 4GB RAM under linux fedora 11.

6.2 Results

In a first set of experiments, algorithms TLS, MQLS, FB-MQLS-D, FB-MQLS-ND are evaluated without dominance rule, only by checking that previously computed cost for a label (j, s', k') is improved at label extension. Table 2 presents the results obtained for these four algorithms. Row *Times* gives the average CPU time for the 100 runs in milliseconds. Row *#Enqueued* displays the average number of nodes enqueued in the heaps during the search. Row *#Visited* provides the average number of visited nodes (scanned successors).

	TLS	MQLS	FB-MQLS-D	FB-MQLS-ND
Times	5294.85	6226.82	5178.89	5376.74
#Enqueued	1 208 870	1 188 140	854 603	880 149
#Visited	2 771 740	2 771 740	2 004 260	2 064 240

Table 2: Comparison of the proposed algorithms without dominance rule

The results show that MQLS algorithm works slowly that TLS although the number of visited label is the same. However, the bidirectional algorithms based on MQLS and deterministic FSA are more efficient than TLS: improvements about 2.18% in terms of CPU time, 29.31% in terms in Enqueued nodes and 27.69% in terms of visited nodes are

experienced. Moreover, the FB-MQLS algorithm based on non-deterministic FSA is less efficient that the FB-MQLS algorithm based on deterministic automaton.

In the second set of experiments, we consider the same algorithms with the basic dominance rule. The results are displayed in Table 3. In each row, we note between parenthesis the relative deviation with the previous results without dominance checks. The basic dominance rule greatly improves all algorithms in terms of CPU time and number of enqueued and visited nodes. TLS, MQLS and FB-MQLS-D algorithms obtain the higher improvement (more than 60%). With the basic dominance rule, MQLS produces less nodes than TLS (about 3% improvement) but it is still slowly (about 5%). But the bidirectional variant of MQML based on deterministic FSA remains the most efficient.

	TLS	MQLS	FB-MQLS-D	FB-MQLS-ND
Times	2000.65 (-62.21%)	2105.47 (-66.19%)	1929.79 (-62.74%)	1991.68 (-62.96%)
#Enqueued	484 120 (-59.95%)	459 116 (-61.36%)	348 016 (-59.28%)	373 791 (-57.53%)
#Visited	1 108 970 (-59.99%)	1 072 390 (-61.31%)	818 587 (-59.16%)	879 154 (-57.41%)

Table 3: Comparison of the proposed algorithms with basic dominance rule

In the third set of experiments, the state-based dominance rule is integrated in all algorithms. Table 4 displays the results and the deviation comparatively to the second set of experiments. The improvement is more important for MQLS and the bidirectional variants (about 12%) than for TLS. The most efficient is still the FB-MQLS-D algorithm. Moreover, with the proposed state-based dominance rule, MQLS algorithm slightly improves the TLS one (less than 2% in terms of CPU time and about 11% in number of nodes.

	TLS	MQLS	FB-MQLS-D	FB-MQLS-ND
Times	1972.88 (-1.39%)	1934.36 (-8,63%)	1707.5 (-11.52%)	1733.3 (-12.97%)
#Enqueued	463 091 (-4.34%)	401 938 (-12.45%)	306 206 (-12.01%)	331 888 (-11.21%)
#Visited	1 061 310 (-4.30%)	936 292 (-12.69%)	717 902 (-12.3%)	778 247 (-11.48%)

Table 4: Comparison of the proposed algorithms with state-based dominance rule

7 Conclusion

We have proposed several algorithms to solve the single-source, single-destination bi-objective multimodal viable shortest path problem where path viability is modeled by a finite state automaton. The considered objectives were the number of transfers and the total travel time. The proposed algorithms are all polynomial in the number of states. In this work, several improvements were brought to the topological Lozano and Storchi algorithm [5] (TLS). We proposed a new multi-queue algorithm (MQLS) for which a bidirectional variant can be easily derived (FB-MQLS). New dominance rules based on the analysis of the finite state automaton were given. In the bidirectional variant, we consider both non-deterministic finite state automaton (which is the reversal of the automaton used in the forward search) and a deterministic variant of this automaton.

An experimental study was carried out on a real-world multimodal network including walk, bus, metro and private vehicle modes. For each problem instance, the set of non-dominated points was found by all algorithms in an short CPU time, allowing their use inside an end-user application which is currently being developped by Mobigis. Algorithms will be integrated in the geographical information system ArcGIS 9.3.1 through which necessary data will be collected and itineraries displayed.

The dominance rules allowed to reduce both the CPU times and the number of visited labels for all algorithms. The most efficient algorithm is the bidirectional one based on the multi-queue concept and the deterministic state automaton.

For futher research, stronger dominance rules could be exhibited, for other special cases of the state automaton.

An extension to multimodality of time-dependent aspects and the related acceleration techniques, such as the one proposed by Nannicini et al [6], is a promosing research direction.

Other multi-objective problems in the multimodal context are of interest and will be the subject of further research, although the complexity of the problem could increase.

References

[1] C. Barett, K. Bisset, M. Holzer, G. Konjevod, M. Marathe, and D. Wagner. Engineering label-constrained shortest-path algorithms. In *4th International Conference on Algorithmic Aspects in Information and Management, AAIM 2008, Shanghai, China*, volume 5034 of *Lecture Notes in Computer Science*, pages 27–37, 2008.

[2] C. Barett, R. Jacob, and M. Marathe. Formal-language-constrained path problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.

[3] M. Bielli, A. Boulmakoul, and H. Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175(3):1705–1730, 2006.

[4] M. Ehrgott. *Multicriteria optimization*, 2nd edition, Springer, 2005.

[5] A. Lozano and G. Storchi. Shortest viable path algorithm in multimodal networks. *Transportation Research Part A : Policy and Practice*, 35(3):225–241, 2001.

[6] G. Nannicini, D. Delling, D. Schultes, and L. Liberti. Bidirectional a* search for time-dependent fast paths. In *7th International Workshop on Experimental algorithms. WEA 2008 Provincetown, MA, USA proceedings*, volume 5038 of *Lecture notes in computer science*, pages 334–346. Springer, 2008.

[7] S. Pallottino and M. G. Scutellà. Shortest path algorithms in transportation models : Classical and innovative aspects. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, pages 245–281. Kluwer Academic Publishers, 1998.

[8] H. D. Sherali and C. Jeenanunta. The approach dependent, time-dependent, label-constrained shortest path problem. *Networks*, 48(2):57–67, 2006.

[9] H.D. Sherali, A.G. Hobeika, and S. Kangwalklai. Time-dependent, label-constrained, shortest path problems with applications. *Transportation Science*, 37(3):278–293, 2003.

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

Appendix

7.1 TLS pseudo-code

Algorithm 1 Topological label-setting algorithm (TLS)

Require: $G(V,E)$, O, D, $d_{ij}, \forall (i,j) \in E$, $k_{\max}$

Set $Q^{now} := \{(O, s_0, 0)\}$, $t^0_{O,s_0} := 0$, $p^0_{O,s_O} := 0$, $t^k_{is} := \infty$, $\forall i \in V \setminus \{O\}, \forall s \in S, \forall k = 0, \ldots, k_{\max}$
Set $Q^{next} := \emptyset$, $lastlabel_{i,s} = \infty$, $\forall i \in V \setminus \{O\}$, $\forall s \in S$
Set $k := 0$
while $Q^{now} \neq \emptyset$ and $k \leq k_{\max}$ **do**
  repeat
    set $(i,s,k) := argmin\{t^k_{js'} | (j,s',k) \in Q^{now}\}$ and set $Q^{now} := Q^{now} \setminus \{(i,s,k)\}$
    if $(i \neq D$ or $s \notin F)$ and $t^k_{is} < lastlabel_{i,s}$ **then**
      set $lastlabel_{is} := t^k_{is}$
      for $j \in FS(i)$ **do**
        set $s' := \delta(m_i, m_j, s)$
        if $s' \neq \emptyset$ and $\forall s'' \preceq s'$, $lastlabel_{j,s''} > t^k_{is} + d_{ij}$ **then**
          if $m_i = m_j$ **then**
            set $t^k_{js'} := t^k_{is} + d_{ij}$, $p^k_{js'} := (i,s,k)$ and $Q^{now} := Q^{now} \cup \{(j,s',k)\}$
            set $lastlabel_{js'} := t^k_{js'}$
          else if $m_i \neq m_j$ and $k + 1 <= k_{\max}$ **then**
            set $t^{k+1}_{js'} := t^k_{is} + d_{ij}$, $p^{k+1}_{js'} := (i,s,k)$ and $Q^{next} := Q^{next} \cup \{(j,s',k+1)\}$
          end if
        end if
      end for
    end if
  until $Q^{now} = \emptyset$ or $(i = D$ and $s \in F)$
  if $i = D$ and $s \in F$, store t^k_{Ds} and p^k_{Ds} (shortest path with k transfers).
  Set $k := k + 1$, $Q^{now} := Q^{next}$ and $Q^{next} := \emptyset$
end while

7.2 MQLS pseudo-code

7.3 FB-MQLS pseudo-code

x, X, $\mathcal{X}$, δ, b are notation symbols instanciated to represent the selected direction where $\mathcal{XQ} \in \{\mathcal{FQ}, \mathcal{BQ}\}$, $XQ_k \in \{FQ_k, BQ_k\}$, $xt^k_{i,s} \in \{ft^k_{i,s}, bt^k_{i,s}\}$, $xp^k_{i,s} \in \{fp^k_{i,s}, bp^k_{i,s}\}$, $\delta(m, m,' s) \in \{\delta^f(m,m',s), \delta^b(m,m',s)\}$, $b_{ij}(t) \in \{t + d_{ij}, t + d_{ji}\}$. The selection of the notation symbol

Algorithm 2 Multi-queue label setting algorithm (MQLS)

Require: $G(V, E)$, O, D, $d_{ij}, \forall (i,j) \in E$, $k_{\max}$

Set $\mathcal{Q} = \{Q_0 := \{(O, s_0, 0)\}\}$, $t^0_{O,s_0} := 0$, $p^0_{O,s_0} := (0, s_0, 0)$, $t^0_{i,s} := \infty$, $\forall i \in V$, $\forall s \in S$, $(i,s) \neq (0, s_0)$
set $K = k_{\max}$
repeat
  set $(i, s, k) := argmin\{t^{k'}_{i',s'} | (i', s', k') \in \mathcal{Q}\}$ and set $Q_k := Q_k \setminus \{(i, s, k)\}$
  if $i = D$ and $s \in F$ **then**
    store $t^k_{i,s}$ and $p^k_{i,s}$ as the shortest path with k transfers. Discard all $Q_{k'}$ with $k' \geq k$. set $K := k - 1$
  else
    for $j \in FS(i)$ **do**
      set $s' := \delta(m_i, m_j, s)$
      if $m_i = m_j$ **then**
        set $k' = k$
      else
        set $k' = k + 1$
      end if
      if $k' \leq K$ and $s' \neq \emptyset$ and $\forall s'' \preceq s'$, $\forall k'' \leq k'$, $t^{k''}_{j,s''} > t^k_{is} + d_{ij}$ **then**
        set $t^{k'}_{j,s'} := t^k_{i,s} + d_{ij}$, $p^{k'}_{j,s'} := (i, s, k)$ and $Q_{k'} := Q_{k'} \cup \{(j, s', k')\}$
      end if
    end for
  end if
until $K < 0$ or $\mathcal{Q} = \emptyset$

is made at steps 7-13. We assume here a deterministic FSA with initial state for the backward pass denoted s_D. SF is the set of forward states while SB is the set of backward states. $delta^f$ is the forward transition function and $delta^b$ is the backward transition function.

Algorithm 3 Bidirectional multi-queue algorithm (FB-MQLS)

Require: $G(V,E)$, O, D, $d_{ij}, \forall (i,j) \in E$, $k_{\max}$ {Initial forward and backward labels}

Set $\mathcal{FQ} = \{FQ_0 := \{(O, s_0, 0)\}\}$, $ft^0_{O,s_0} := 0$, $fp^0_{O,s_0} := (0, s_0, 0)$, $ft^0_{i,s} := \infty$, $\forall i \in V$, $\forall s \in SF$, $(i,s) \neq (0, s_0)$

Set $\mathcal{BQ} = \{BQ_0 := \{(D, s_D, 0)\}\}$, $bt^0_{O,s_D} := 0$, $fp^0_{O,s_D} := (D, s_D, 0)$. Set $bt^0_{i,s} := \infty$, $\forall i \in V$, $\forall s \in SB$, $i \neq 0$ or $s \neq s_D$.

set $K = k_{\max}$

{Main Loop: start by getting minimum time label among all priority queues}

repeat

  Let $(i^f, s^f, k^f) := argmin\{ft^{k'}_{i',s'} | (i', s', k') \in \mathcal{FQ}\}$

  Let $(i^b, s^b, k^b) := argmin\{bt^{k'}_{i',s'} | (i', s', k') \in \mathcal{BQ}\}$

  {Direction setting}

  if $ft^{k^f}_{i^f,s^f} \leq bt^{k^b}_{i^b,s^b}$ **then**

    set $(i, s, k) := (i^f, s^f, k^f)$

    set $x = f$, $X = F$, $\mathcal{X} = \mathcal{F}$, $\overline{x} = b$, $\delta = \delta^f$, $b_{ij}(t) = t + d_{ij}$

  else

    set $(i, s, k) := (i^b, s^b, k^b)$

    set $x = b$, $X = B$, $\mathcal{X} = \mathcal{B}$, $\overline{x} = f$, $\delta = \delta^b$, $b_{ij}(t) = t + d_{ji}$

  end if

  $XQ_k := XQ_k \setminus \{(i, s, k)\}$

  {Extension checking}

  for $j \in XS(i)$ **do**

    set $s' := \delta(m_i, m_j, s)$

    if $m_i = m_j$ **then**

      set $k' = k$

    else

      set $k' = k + 1$

    end if

    if $k' \leq K$ and $s' \neq \emptyset$ and $b_{ij}(xt^k_{i,s}) < xt^{k'}_{j,s'}$, $\forall k'' \leq k'$, $s'' \preceq s'$ **then**

      set $xt^{k'}_{j,s'} := b_{ij}(xt^k_{i,s})$, $xp^{k'}_{j,s'} := (i, s, k)$ and $XQ_{k'} := XQ_{k'} \cup \{(j, s', k')\}$

      {Connection checking}

      set minconnection:= ∞; set $k^* = -1$

      for $(j, s', k'') \in \overline{\mathcal{X}}\mathcal{Q}$, $k' + k'' \leq K$ **do**

        if $L^{k'+k''} > xt^{k'}_{j,s'} + \overline{x}t^{k''}_{j,s'}$ **then**

          set $L^{k'+k''} := xt^{k'}_{j,s'} + \overline{x}t^{k''}_{j,s'}$

          if $L^{k'+k''} <$minconnection **then**

            set minconnection:= $L^{k'+k''}$; set $k^* := k' + k''$

          end if

        end if

      end for

    end if

  end for

  {Opimality test}

  if $k^* \neq -1$ and $L^{k^*} \leq ft^{k^f}_{i^f,s^f} + bt^{k^b}_{i^b,s^b}$ **then**

    store L^{k^*} as the shortest path with k^* transfers. Discard all $FQ_{\tilde{k}}$ and $BQ_{\tilde{k}}$ with $\tilde{k} \geq k^*$. set $K := k^* - 1$

  end if

until $K < 0$ or $\mathcal{FQ} = \mathcal{BQ} = \emptyset$