# Label-setting algorithms for a polynomial bi-objective multimodal shortest path problem

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

# Label-setting algorithms for a polynomial bi-objective multimodal 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 Cedex 4, France
[2]Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, 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. In this paper, such constraints are gathered under the concept of viable path modeled by a deterministic finite state automaton. Several polynomial algorithms are proposed to solve a bi-objective problem proposed in [13], where the goal is to find all the non-dominated viable shortest paths under the two objectives "travel time" and "number of modal transfers". Among these algorithms, we consider an improved variant of the topological label-setting algorithm provided by Lozano and Storchi [13], a new multi-label multi-queue algorithm and its bidirectional variant. The different algorithms are compared on a real network. The results show that, on the considered network, the proposed algorithms outperform the Lozano and Storchi [13] algorithm, both for the time-independent and time-dependent case. Finally, $A^*$ acceleration techniques are discussed.
**keywords:** bi-objective viable shortest paths, multimodal transportation, finite state automaton, label-setting algorithms, bidirectional search, $A^*$

## 1 Introduction

Computing shortest paths in the context of monomodal passenger transportation, where a single transportation mode (e.g. private vehicle, bus, metro) is used during the passenger's itinerary, was subject to extensive research since the publication of Dijkstra's algorithm in the 1950s. Among the considered extensions of the basic shortest path problems, the case where travel times (or costs) are time-dependent, allowing to take into account public transportation timetables or traffic congestion hours, has also been widely studied. Nowadays, thanks to powerful acceleration and/or preprocessing techniques (such as bidirectional search, $A^*$ search, landmarks, contraction hierarchies, etc.), computing shortest-paths very fastly in large-scale network, either in a time-independent or in a time-dependent context, is not a challenge anymore [7, 8, 17]. Note that the best results obtained consider the FIFO assumption for the time-dependent network, i.e. for an arc $(i, j)$ departing earlier from $i$ implies arriving earlier at $j$.

The case of multimodal passenger transportation, which lies in the transportation of one or several passengers with diffferent modes during the same itinerary, has been much less adressed. However, multimodal transportation is subject to a growing interest in the research community, as multimodality is now widely accepted for urban transportation as a necessary alternative to the exclusive use of private vehicles. In this paper, we consider a polynomial bi-objective shortest path problem in a multimodal urban transportation network: the minimum time/minimum number of transfers multimodal viable shortest-path problem. We consider a time-independent problem variant initially proposed by Lozano and Sorchi [13] denoted as `BI-MM-V-SPP` and a time-dependent extension (under the FIFO assumption), denoted `BI-TD-MM-V-SPP`. We also consider an application to the urban area of Toulouse (France) the purpose of which being to evaluate the tractability of the considered problem on real-life network.

In Section 2 we briefly recall the existing litterature on the multimodal shortest path problem, so as to introduce our motivation for the present study. Section 3 formally defines the problem. The proposed algorithm for `BI-MM-V-SPP` and `BI-TD-MM-V-SPP`, i.e., an improved variant of the topological label-setting algorithm provided by Lozano and Storchi [13], a new multi-label multi-queue algorithm and its bidirectional variant, are presented in Section 4. Their performance on the considered real network are compared in Section 5. For further speedups, integration of goal-oriented ($A^*$) techniques in the algorithms and associated additional computational experiments are discussed in Section 6. Concluding remarks are drawn in Section 7.

## 2 Literature review

Multimodal transportation raises network modeling issues [6, 12]. A simple way of modeling the network, used by many authors [16, 13, 4], lies in assuming that the set of nodes is partitionned according to the modes. Arcs linking two nodes of differents subsets is called a transfer arc. Equivalently, nodes and/or arcs are labeled according to the associated mode [2]. Once such a network is defined, one typically seeks to model the fact that some sequence of modes constituting a path can be infeasible in practice. A first (relaxed) way of taking account of such mode restrictions for shortest path computations was proposed by Modesti and Sciomachen [16], who proposed a extension of Dijkstra's algorithm to minimize a (single) global utility function defined by a weighted sum of modal characteristics of a path (time spent on the private car, time spent on the bus or subway, walking time, waiting time,...). For shortest path computation including hard modal constraints, (possibily infinite) mode-dependent travel times were used by Ziliaskopoulos and Wardell [23], together with an arc representation allowing to design mode constraints involving three nodes. A more general way of modeling the multimodal constraints was proposed by Barett *et al.* [2]. Each mode being viewed as an element of an alphabet, each arc of the network being labeled by a mode, the mode restrictions can be described by a regular language over the alphabet. The multimodal shortest path problem then amounts to a regular language-constrained shortest path problem. As a regular language can be represented by a non-deterministic finite state automaton (NFA), Barett *et al.* [2] proved that the problem is polynomial in the number of states of the automaton. In [1, 22, 21], practical implementation issues of this method are discussed. Barett *et al.* [1] proposed $A*$ and bidirectional accelerations. Considering deterministic finite-state automaton (DFA) as input, Sherali *et al.* [21] extend the problem to time-dependence and propose a strongly polynomial algorithm for FIFO graphs. Sherali and Jeenanunta [22] 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.

The main drawback of the approaches based on the regular language constraint shortest path problem are that they all consider only a single objective. However, when several modes are available a user may want to select her/his itinerary among a set of alternatives, taking account of several objectives. Two general classes of multi-objective multimodal shortest path problems have already been considered in the litterature: namely, polynomial problems and NP-hard problems.

The simplest polynomial bi-objective multimodal problem was introduced by Pallottino and Scutellà [20]. They considered the problem to find all non-dominated point-to-point paths considering the "minimum time" and "minimum number of transfers" objectives, a transfer being defined as a mode change along the path. No multimodal restrictions were considered. They proposed a polynomial topological label-setting algorithm to solve the problem. In [13], Lozano and Storchi directly extend this problem and the topological algorithm to integrate mode restrictions using a DFA in a time-independent network, defining the `BI-MM-V-SPP` considered in the present paper.

Bielli *et al* [4] consider a simplified version of the DFA model but include time-dependent arcs and time penalties for turning movements. Their problem can be considered as a variant of the `BI-TD-MM-V-SPP`. Their 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 [20] algorithm, with labels on arcs. Experimental validations for the `BI-MM-V-SPP` and the `BI-TD-MM-V-SPP` are limited to small networks. The largest one, presented in [4], 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 the `BI-MM-V-SPP` and the `BI-TD-MM-V-SPP`. The main purpose of the paper is to study the tractability of solving these two bi-objective problems on a real network in reasonable computational time.

A more general class of problems consider general objective function and mostly propose extensions to multimodality of the NP-hard bi-objective shortest path problem. Although we focus in this paper on the polynomial `BI-MM-V-SPP` and `BI-TD-MM-V-SPP`, we mention a recent experimental study carried out for a more general problem since the number of transfers and minimum time objectives were also considered among other objectives. Gräbener *et al.* [10] present an extension of Martin's algorithm [15] to deal with the multimodal, multiobjective shortest path, considering only basic mode restrictions (the finite state automaton formalism is not used). When only the minimum number of transfers and minimum time objectives are considered, the method shows very fast computational times. For three modes (cycling, walking and public transportation), in a time-dependent context, the pareto-optimal paths are computed in 71.9 milliseconds in average for a network of 36694 nodes and 171443 edges. However the number of non-dominated path is on average equal to 1.2. Actually, since the cycling mode can be taken from the origin to the destination (or left anywhere in the network), it generally dominates the other modes. This recent study partially answered our question in the sense that they showed that the `BI-TD-MM-V-SPP` is actually tractable when no complex mode restrictions are defined and when one of the modes tends to dominate the others.

In this paper we propose specific algorithms for the `BI(-TD)-MM-V-SPP` and we evaluate the practical tractability of real instances admitting significantly more non-dominated solutions and where complex mode restrictions are represented by a finite state automaton.

# 3 Problem statement

## 3.1 `BI-MM-V-SPP` and `BI-TD-MM-V-SPP` definition

Let $M$ denotes the set of modes. The multimodal transportation network is modeled by a multi-layer graph $G(V, E)$ such that each layer corresponds to a mode $m \in M$. A mode $m_i \in M$ is defined for each node $i \in V$. For the `BI-MM-V-SPP`, a travel time $d_{ij}$ is associated to each arc $(i, j) \in E$. For the `BI-TD-MM-V-SPP`, a function $a_{ij}(t)$ is associated to each arc $(i, j) \in E$. It gives the arrival time to node $j$ given that departure time from $i$ is $t$. An arc $(i, j)$ such that $m_i \neq m_j$ is called a **transfer arc**.
In terms of multimodal characteristics, each path in $G$ yields by a sequence (or string) of modes. Among all strings of modes, only a subset of strings are acceptable according to a passenger's preferences or to feasibility constraints. The acceptable mode sequences are represented via a deterministic finite state automaton (DFA). The input DFA 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 $\delta : M \times M \times S \to S$ 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 $m_j = m'$. We assume that $\delta(m, m', s) = \emptyset$ denotes the case where the transition is unfeasible.

A **viable path** is a path in $G$ from an origin node $O$ to the destination node $D$ verifying the constraints represented by the DFA. 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 [9] 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.

## 3.2 DFA example

For our experimental evaluation on a real network, we consider the case where $M = \{wa, bu, pr, me\}$ (walking, bus, private car, metro). The considered intineraries are assumed to be from home to another place. Hence, the viability constraints taking into account are the following: the private car can be taken only from $O$ and, once left, cannot be taken again. Moreover, the private car can only be left at a subset of nodes representing parking places. We assume for the considered example that the private car cannot be left at the destination. For the metro, one assume that it can be taken at any time but, once left, cannot be taken again. We assume also that the origin and destination nodes are not in the metro.

The deterministic finite state automaton with $|S| = 5$ represented in the Figure 1 models these viability constraints on both metro and private car. Transition arcs between

states are labeled by a mode $m \in M$ where $M = \{wa, bu, pr, me\} \cup \{m_O\}$, $m_O$ being a fictive mode labeling only the origin $O$. A transition from state $s$ to state $s'$ labeled by $m \in M$ describes 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 $s_0$. In Figure 1 state $s_1$ means that private car was not taken at $O$ and so mode $pr$ is

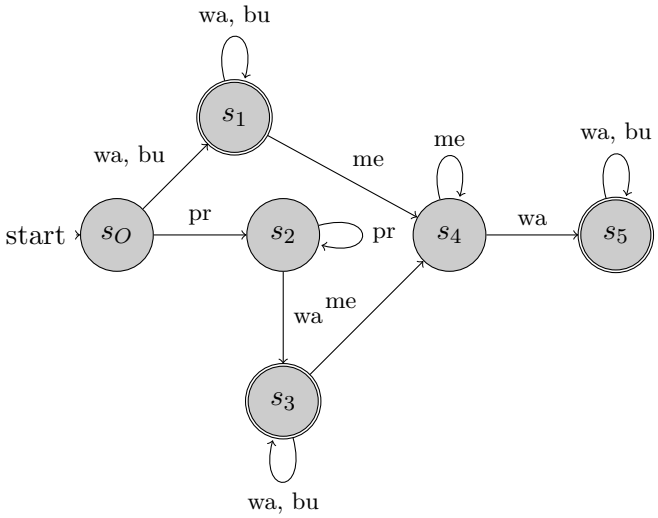



Figure 1: Original deterministic finite state automaton

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 car 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 double circle 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.

# 4 Algorithms for the `BI-MM-V-SPP` and the `BI-TD-MM-V-SPP`

In this section, we detail three algorithms to solve both the `BI-MM-V-SPP` and the `BI-TD-MM-V-SPP`. All algorithms are based on a label setting principle which is described in Section 4.1. All algorithms also use dominance rules which are presented in Section 4.2. The first algorithm (TLS), a label-setting version of the topological label setting algorithm proposed by [13], is given in Section 4.3. The second algorithm (MQLS), described in Section 4.4 is a new label setting algorithm based on multiple priority queues. Section 4.5 presents the third algorithm (FB-MQLS), a bidirectional (Forward-Backward) adaptation of MQLS.

We decribe algorithms TLS and MQLS in the most general context, i.e. the time-dependent one. In this context, a travel time function $a_{ij}(t)$ is associated to each arc $(i, j)$. The time-independent case can be obtained by setting $a_{ij}(t) = t + d_{ij}$. We assume the FIFO assumption holds, i.e. if $t_1 \leq t_2$ then $a_{ij}(t_1) \leq a_{ij}(t_2)$ for each arc $(i, j) \in E$. Assuming

time discretization, for each time-dependent arcs $(i, j)$, arrival times at destination are store in a table $A_{ij}^{\delta}$ where $\delta \in \Delta_{ij}$ and $\Delta_{ij}$ is a set of intervals. Each interval $\delta \in \Delta_{ij}$ is defined by bounds $S_\delta$ and $F_\delta$ such that if $t \in \{S_\delta, \ldots, F_\delta - 1\}$, $a_{ij}(t) = A_{ij}^{\delta}$. In other words $a_{ij}(t)$ is piecewise constant. Hence, given $t$, $a_{ij}(t)$ can be computed in $O(1)$. It follows that algorithms TLS and MQLS described below have the same worst case time complexity for `BI-MM-V-SPP` and the `BI-TD-MM-V-SPP`. The bidirectional algorithm FB-MQLS however is first decribed for solving `BI-MM-V-SPP` and then extended to `BI-TD-MM-V-SPP`.

### 4.1 Label setting principle

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 predecessor label of $(i, s, k)$ on the path. When $p_{is}^k = (j, s', k')$ it means that (a) arc $(j, i)$ is used on the path and that (b) the state of the path on $i$ 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.

All the proposed algorithms implement differently the following basic principles. 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 $Q$. 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 $Q$ are scanned. For each successor $j$, we first check if taking arc $(i, j)$ is viable 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 then 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 label is inserted in $Q$ if some dominance rules do not apply. Otherwise the label is discarded.

### 4.2 Dominance rules and state reduction

In this section we give dominance rules allowing to discard labels. A first dominance rule, 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 $t_{is}^k \leq t_{is}^{k'}$, then the label $(i, s, k')$ can be discarded (because it is dominated by the label $(i, s, k)$).*

Obviously, under the described conditions, any O-D path issued from $(i, s, k)$ is non-dominated by any O-D path issued from $(i, s, k')$.

A second dominance rule, 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

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

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

From the state-based dominance rule, we also derive the following state merging conditions that can be used to reduce the initial automaton presented in 1 and initially proposed by Lozano and Storchi [13]. Such a reduction is beneficial for the computational requirements as the next sections show that the complexity of the algortithms depends on the number of states of the DFA.

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

We remark in Figure 1 that states $s_1$ and $s_3$ verify the above-described condition and then, we consider the reduced automaton of the Figure 2.

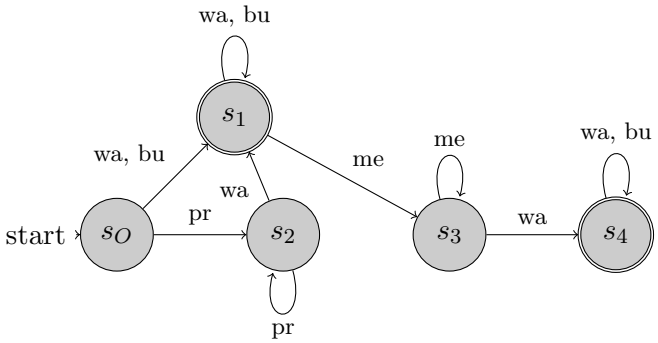



Figure 2: Reduced deterministic finite state automaton

### 4.3 Topological label-setting (TLS) algorithm

The topological Pallottino and Scutellà [20] algorithm was extended by Lozano and Storchi algorithm [13] to path viability modeled by a DFA. We describe below a label-setting variant of algorithm TLS (the original algorithm of [13] being described as a label-correcting algorithm).

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 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 label $(j, s', k')$, we have only to keep track of the shortest time found so far to reach $(j, s', k'')$ with $k'' \leq k'$, denoted $lastlabel_{js'}$ and the label is dominated if $t^{k'}_{js'} \geq lastlabel_{js'}$, as the previously encountered label cannot have more transfers. The complexity of the state-based dominance rule is in $O(|S|)$ : a given label $(j, s', k')$ obtain by extension of a label $(i, s, k)$ is dominated if $lastlabel_{j,s''} > a_{ij}(t^k_{is})$ for all states $s''$ such that $s'' \preceq s'$.

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

**Algorithm 1** Topological label-setting algorithm (TLS)

**Require:** Graph $G(V, E)$, DFA $A$, $O$, $D$, $a_{ij}(.), \forall (i, j) \in E$, $k_{\max}$

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 $lastlabel_{is} := t^k_{is}$
      {Scan successors of label $(i, s, k)$}
      **for** $j \in FS(i)$ **do**
        set $s' := \delta(m_i, m_j, s)$
        **if** $s' \neq \emptyset$ and $\forall s'' \preceq s'$, $lastlabel_{j,s''} > a_{ij}(t^k_{is})$ **then**
          **if** $m_i = m_j$ **then**
            set $t^k_{js'} := a_{ij}(t^k_{is})$, $p^k_{js'} := (i, s, k)$ 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 $t^{k+1}_{js'} := a_{ij}(t^k_{is})$, $p^{k+1}_{js'} := (i, s, k)$ 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 $s \in F$ **then**
    store $t^k_{Ds}$ and $p^k_{Ds}$ (shortest path with $k$ transfers).
  **end if**
  Set $k := k + 1$, $Q^{now} := Q^{next}$ and $Q^{next} := \emptyset$
**end while**

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 can be done in

$O(\log(n|S|))$. Successor scan with the basic dominance rule (in $O(1)$) and possible insertion (in $O(\log(n|S|))$ has a worst-case complexity in $O(|FS_i|\log(n|S|))$ where $FS_i$ is the set of direct successors of $i$. The complexity of operation (a) is ignored (neglected) since it is lower than the complexity of operation (b). 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 $O(k_{max}|S||E|(|S|+\log(n|S|)))$.

To illustrate the TLS algorithm behavior, consider the bi-objective multimodal shortest path problem from node $x_1$ to node $x_5$, represented in Figure 3 (there are no multimodal restrictions so a label corresponds to a pair $(i,k)$). There are 2 modes and 5 nodes (transfer arcs are represented by dashed arcs) 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

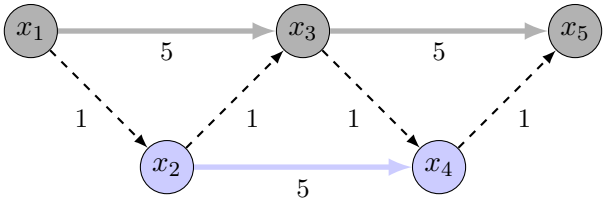



Figure 3: Small example of multimodal problem

TLS.
**Iteration 1:** $k=0$, $Q^{now}=\{x_1\}$, $t_1^0=0$
$(i,k)=(x_1,0)$:
$(j,k')=(x_3,0)$, $t_3^0=5$, $Q^{now}=\{x_3\}$
$(j,k')=(x_2,1)$, $t_2^1=1$, $Q^{next}=\{x_2\}$
$(i,k)=(x_3,0)$:
$(j,k')=(x_5,0)$, $t_5^0=10$, $Q^{now}=\{x_5\}$
$(j,k')=(x_4,1)$, $t_4^1=6$, $Q^{next}=\{x_2,x_4\}$
$(i,k)=(x_5,0)$: destination reached: shortest path with 0 transfer
**Iteration 2:** $k=1$, $Q^{now}=\{x_2,x_4\}$, $t_2^1=1$, $t_4^1=6$, $Q^{next}=\emptyset$
$(i,k)=(x_2,1)$:
$(j,k')=(x_4,1)$, $t_4^1=6$, this label is dominated
$(j,s')=(x_3,2)$, $t_3^2=2$, $Q^{next}=\{x_3\}$
$(i,k)=(x_4,1)$:
$(j,k')=(x_5,2)$, $t_5^2=7$, $Q^{next}=\{x_3,x_5\}$
stop: $Q^{now}=\emptyset$
**Iteration 3:** $k=2$, $Q^{now}=\{x_3,x_5\}$, $t_3^2=2$, $t_5^2=7$, $Q^{next}=\emptyset$
$(i,k)=(x_3,2)$:
$(j,k')=(x_5,2)$, $t_5^2=7$, this label is dominated
$(j,k')=(x_4,3)$, $t_4^3=3$, $Q^{next}=\{x_4\}$
$(i,k)=(x_5,7)$: destination reached: shortest path with 2 transfers
**Iteration 4:** $k=3$, $Q^{now}=\{x_4\}$, $t_4^3=3$, $Q^{next}=\emptyset$
$(i,k)=(x_4,3)$:
$(j,k')=(x_5,4)$, $t_5^4=4$, $Q^{next}=\{x_5\}$

stop : $Q^{now} = \emptyset$
**Iteration 5:** $k = 4$, $Q^{now} = \{x_5\}$, $t_5^4 = 4$, $Q^{next} = \emptyset$
$(i, k) = (x_5, 4)$: destination reached: shortest path with 4 transfers
**Iteration 6:** $k = 5$, $Q^{now} = \emptyset$, stop

Then, in this example, three non-dominated solutions are generated for the bi-objective shortest path from $x_1$ to $x_5$: the first solution with 0 transfer and a travel time equals to 10, the second solution corresponds to 1 transfer and a travel time of 7 and the third one leads to 4 transfers and a travel time equals to 4. The algorithm stops because there is no more label in $Q_{now}$ even if the limit on the maximal number of transfers is not reached. The TLS algorithms generates non-dominated solutions in increasing order of the number of transfers and decreasind order of travel time.

### 4.4 Multi-queue label-setting (MQLS) algorithm

We propose an alternative multi-queue algorithm that computes the shortest paths in increasing order of the time criterion values and then in decreasing order of the number of transfers. Instead of considering two queues $Q^{now}$ 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 $Q_{k'}$. The algorithm stops when the shortest path with 0 transfer is found or when all queues are emptied. The algorithm pseudo code is given in (Algorithm 2). Now we determine the algorithm complexity, using binary heaps for each $Q_k \in \mathcal{Q}$. In $\mathcal{Q}$ at most $k_{\max} n|S|$ labels are stored and dequeued (**marked**). For each iteration, there are three operations: (a) search of the minimum time value in the $k_{\max}$ queues; (b) deletion of the corresponding label in $O(\log(n|S|))$; and (c) for each scanned successor, a dominance check is possibly followed by an insertion operation in the appropriate queue in $O(\log(n|S|))$. 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, with this dominance rule, the operation (c) as an $O(|FS_i|(k_{\max} + \log(n|S|))$ worst-case time complexity. The state-based dominance rule can be applied in $O(k_{max}|S|)$, then the operation (c) as a worst-case time complexity of $O(|FS_i|(k_{\max}|S| + \log(n|S|))$.

Taking account of (a) and (b) operations, with the basic dominance rule we obtain a worst-case complexity of

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

and with the state-based dominance rule this worst-case complexity is

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

The worst-case time complexity is increased compared to the TLS algorithm by a $k_{\max}$ factor.
Considering the example of Figure 3, we show hereafter the step-by-step execution of this

**Algorithm 2** Multi-queue label setting algorithm (MQLS)

**Require:** Graph $G(V, E)$, DFA $A$, $O$, $D$, $a_{ij}(.), \forall (i,j) \in E$, $k_{\max}$

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 $\forall s'' \preceq s'$, $\forall k'' \leq k'$, $t^{k''}_{j,s''} > a_{ij}(t^k_{is})$ **then**
        set $t^{k'}_{j,s'} := a_{ij}(t^k_{i,s})$, $p^{k'}_{j,s'} := (i, s, k)$ and $Q_{k'} := Q_{k'} \cup \{(j, s', k')\}$
      **end if**
    **end for**
  **end if**
**until** $K < 0$ or $\mathcal{Q} = \emptyset$

algorithm.
**Iteration 1:** $Q_0 = \{x_1\}$, $t^0_1 = 0$
$(i,k) = (x_1, 0)$:
$(j,k') = (x_3, 0)$, $t^0_3 = 5$, $Q_0 = \{x_3\}$
$(j,k') = (x_2, 1)$, $t^1_2 = 1$, $Q_1 = \{x_2\}$
**Iteration 2:** $Q_0 = \{x_3\}$, $t^0_3 = 5$, $Q_1 = \{x_2\}$, $t^1_2 = 1$
$(i,k) = (x_2, 1)$:
$(j,k') = (x_4, 1)$, $t^1_4 = 6$, $Q_1 = \{x_4\}$
$(j,k') = (x_3, 2)$, $t^2_3 = 2$, $Q_2 = \{x_3\}$
**Iteration 3:** $Q_0 = \{x_3\}$, $t^0_3 = 5$, $Q_1 = \{x_4\}$, $t^1_4 = 6$, $Q_2 = \{x_3\}$, $t^2_3 = 2$
$(i,k) = (x_3, 2)$:
$(j,k') = (x_5, 2)$, $t^2_5 = 7$, $Q_2 = \{x_5\}$
$(j,k') = (x_4, 3)$, $t^3_4 = 3$, $Q_3 = \{x_4\}$
**Iteration 4:** $Q_0 = \{x_3\}$, $t^0_3 = 5$, $Q_1 = \{x_4\}$, $t^1_4 = 6$, $Q_2 = \{x_5\}$, $t^2_5 = 7$, $Q_3 = \{x_4\}$, $t^3_4 = 3$
$(i,k) = (x_4, 3)$:
$(j,k') = (x_5, 4)$, $t^4_5 = 4$, $Q_4 = \{x_5\}$
**Iteration 5:** $Q_0 = \{x_3\}$, $t^0_3 = 5$, $Q_1 = \{x_4\}$, $t^1_4 = 6$, $Q_2 = \{x_5\}$, $t^2_5 = 7$, $Q_3 = \emptyset$, $Q_4 = \{x_5\}$, $t^4_5 = 4$
$(i,k) = (x_5, 4)$: destination reached, shortest path with 4 transfers, $K = 3$
**Iteration 6:** $Q_0 = \{x_3\}$, $t^0_3 = 5$, $Q_1 = \{x_4\}$, $t^1_4 = 6$, $Q_2 = \{x_5\}$, $t^2_5 = 7$, $Q_3 = \emptyset$, $K = 3$
$(i = x_3, 0)$:

$(j, k') = (x_5, 0)$, $t_5^0 = 10$, $Q_0 = \{x_5\}$
$(j, k') = (x_4, 1)$, $t_4^1 = 6$: is dominated
**Iteration 7:** $Q_0 = \{x_5\}$, $t_5^0 = 10$, $Q_1 = \{x_4\}$, $t_4^1 = 6$, $Q_2 = \{x_5\}$, $t_5^2 = 7$, $Q_3 = \emptyset$, $K = 3$
$(i, k) = (x_4, 1)$
$(j, k') = (x_5, 2)$, $t_5^2 = 7$: is dominated
**Iteration 8:** $Q_0 = \{x_5\}$, $t_5^0 = 10$, $Q_1 = \emptyset$, $Q_2 = \{x_5\}$, $t_5^2 = 7$, $Q_3 = \emptyset$, $K = 3$
$(i, k) = (x_5, 2)$: destination reached, shortest path with 2 transfers, $K = 1$
**Iteration 8:** $Q_0 = \{x_5\}$, $t_5^0 = 10$, $Q_1 = \emptyset$, $K = 1$
$(i, k) = (x_5, 0)$ : destination reached, shortest path with 0 transfer, $K = -1$
**Iteration 9:** $K = -1$ ($\mathcal{Q} = \emptyset$): stop

This example illustrates than MQLS algorithm obtains solution in decreasing order of the number of transfers : the first obtained solution is the one with 4 transfers, the second has 2 transfers and the third one has 0 transfer.

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 bi-objective $O-i$ 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.

## 4.5 Bidirectional Multi-Queue Label Setting Algorithm (FB-MQLS)

We propose an adaptation of MQLS in a bidirectional way taking advantage of the multi-queue characteristics. 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 $s$.

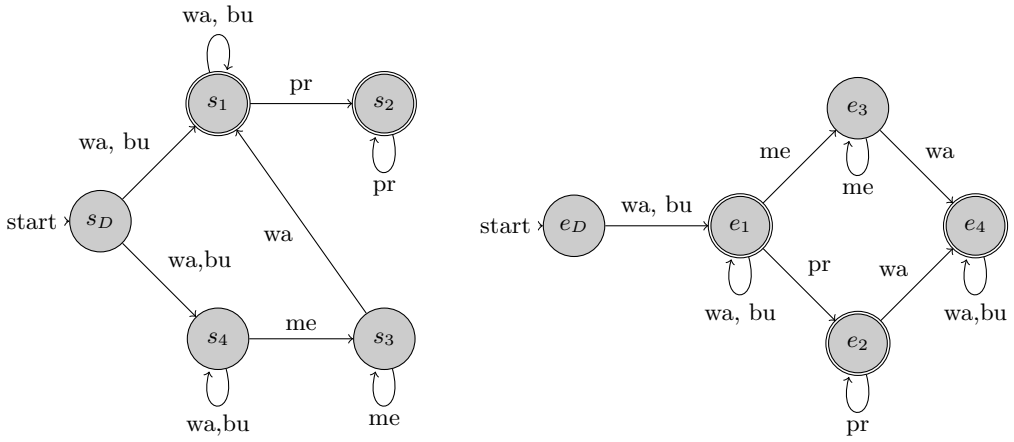
As already mentionned, we first describe FB-MQLS in a time-independent context (i.e. for the `BI-MM-V-SPP`). There are two main issues in designing a bidirectional algorithm for the considered multimodal problem. The first issue addressed in Section 4.5.1, consists in modeling backward path viability. The second issue for designing a bidirectional algorithm for our problem lies in exploiting the connection between a forward and backward label in the bi-objective context. The multi-queue structure is then fully exploited as several label queues may be discarded when a connection meets the condition described in Section 4.5.2. These issues being adressed, the algorithm FB-MQLS is described in Section 4.5.3 for the `BI-MM-V-SPP`. Extension to the `BI-TD-MM-V-SPP` is discussed in Section 4.5.4.

### 4.5.1 Modeling backward path viability

We exhibit below two different possibilities to model backward path viability. Let $FA = (S^{FA}, M, \delta^{FA}, s_0^{FA}, F^{FA})$ denotes the automaton for the forward search and $BA = (S^{BA}, M, \delta^{BA}, s_0^{BA}, F^{BA})$ the automaton for the backward search. To obtain $BA$ from $FA$, the

first possibility is simply to reverse the arcs of $FA$. Generally, the obtained state automaton is non-deterministic (see left part of Figure 4). In this figure, the initial 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 $\delta^{BA}(m_i, m_j, s)$ gives a set of possible states. For example $\delta^{BA}(m_D, wa, s_5) = \{s_1, s_4\}$ (where $m_D$ denotes the mode at the destination, ie. either $wa$ either $bus$). This means what when arriving by walk at the destination, it could be that the metro was taken (state $s_4$) or was not taken (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 indeterminism 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, however one can not ensure that for a given non-deterministic automaton with $|S|$ states, the equivalent deterministic automaton has less than $2^{|S|}$ states. An issue then is to generate the deterministic automaton with a minimal number of states. We display in right part of Figure 4 a possible deterministic finite state automaton for $BA$ that we have obtained manually by a logical description of a reverse viability of a path, involving the same number of states. Between these two deterministic automaton, $CS^{BA \to FA}$ (respectively $CS^{FA \to BA}$) denotes the set of $FA$ (resp. $BA$) states compatible with a given state of $BA$ (resp. $FA$).




(a) Non deterministic automata

(b) Deterministic automata

Figure 4: Automata for the backward search

### 4.5.2 Connection and queue discarding rule

The second issue for designing a bidirectional algorithm for the considered problem 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 minimal connection is made between a label $ft^h_{i,s^f}$ and a label $bt^q_{i,s^b}$ such that the state $s^f$ of $FA$ is compatible with state $s^b$ of $BA$, if condition

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

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

### 4.5.3 Algorihm description and complexity

Algorithm FB-MQLS pseudo-code is given in (Algorithm 3). $\mathcal{FQ}$ is initalized to a single priority queue $FQ_0$ with a single label $(O, s_0, 0)$ and $\mathcal{BQ}$ is initialized to a single priority queue $BQ_0$ 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 $BQ_k$) and the forward or backward extension is performed in a similar way as in Algorithm MQLS (see Algorithm 4 for the forward extension, the backward extension algorithm being symmetrical). The only difference is that (for the forward extension), for each new label $(j, s', k')$, a connection with the opposite direction search is searched by scanning all labels $(j, s^b, k'')$ with $k' + k'' \leq K$ and $s_b \in CS^{FA \to BA}(s')$ which possibly yield an $O - D$ path of less than $K$ transfers. If such a connection is established, its path time ($ft^{k'}_{j,s'} + bt^{k''}_{j,s^b}$) 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, the queue discarding rule can be tested by comparing the minimum time $L^{k^*}$ obtained among the extensions, with the 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.

Compared to MQLS, there is a computational overhead (only at label extension) induced by connection search (step 10 of Algorithm 4). Let us examine first this overhead in terms of complexity for the forward extension. With the basic dominance rule, recall the complexity of label extension is $O(|FS_i|(k_{\max} + \log(n|S|))$ for MQLS. For a backward deterministic automaton, connection search introduces an additive term equal to $k_{\max}$ to search for all labels $(j, s^b, k'')$ as there is a single compatible state. For the non-deterministic backward automaton, the additive term can be bounded from above by $k_{\max}\tilde{s}$ where $\tilde{s} = \max_{s \in S^{FA}} |CS^{FA \to BA}(s)| \leq |S^{BA}|$. Hence for the basic dominance rule, the complexity of the forward extension (Algorithm 4)) is $O(|FS_i|(k_{\max} + \log(n|S^{FA}|))$ for the deterministic backward automaton and $O(|FS_i|(k_{\max}\tilde{s} + \log(n|S^{FA}|))$ for the non-deterministic backward automaton. For the state based dominance rule, the complexity of forward extension is $O(|FS_i|(k_{\max}|S^{FA}| + \log(n|S^{FA}|))$ for the deterministic bakward automaton and $O(|FS_i|(k_{\max}(|S^{FA}| + \tilde{s}) + \log(n|S^{FA}|))$ for the non-deterministic backward automaton. Since the forward automaton is assumed to be deterministic, the complexity of the backward extension is always equal to $O(|BS_i|(k_{\max} + \log(n|S^{BA}|))$ for the basic dominance rule and to $O(|FS_i|(k_{\max}(|S^{BA}|) + \log(n|S^{BA}|))$ for the state-based dominance rule.

We show below the execution of this algorithm for the example of Figure 3.

$FQ_0 = \{x_1\}$, $ft^0_1 = 0$, $BQ_0 = \{x_5\}$, $bt^0_5 = 0$, $k^* = -1$, $K = 5$

**Iteration 1:** select $x_1$ from $FQ_0$, forward search, $(i, k) = (x_1, 0)$

$(j, k') = (x_3, 0)$: $ft^0_3 = 5$, $FQ_0 := \{x_3\}$, no connection

$(j, k') = (x_2, 1)$: $ft^1_2 = 1$, $FQ_1 := \{x_2\}$, no connection

**Iteration 2:** select $x_5$ from $BQ_0$, backward search, $(i, k) = (x_5, 0)$

$(j, k') = (x_3, 0)$: $bt^0_3 = 5$, $BQ_0 := \{x_3\}$, 1 connection: $ft^0_3 = 5 \Rightarrow L^0 = 10$, $k^* = 0$

$(j, k') = (x_4, 1)$: $bt^1_4 = 1$, $BQ_1 := \{x_4\}$, no connection

$argmin\{ft^{k'}_{i',s'} | (i', s', k') \in \mathcal{FQ}\} = (x_2, 1)$, $ft^1_2 = 1$

$argmin\{bt^{k'}_{i',s'} | (i', s', k') \in \mathcal{BQ}\} = (x_4, 1)$, $bt^1_4 = 1$

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

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

{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 $s \neq s_D$.

{Initial minimal connection}

set $K = k_{\max}$ and set $L^k = \infty, \forall k, 0 \leq k \leq k_{\max}$

{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 $(i,s,k) := (i^f, s^f, k^f)$ and set $FQ_k := FQ_k \setminus \{(i^f, s^f, k^f)\}$

    **if** $i = D$ and $s \in F^{FA}$ **then**

      set $L^k := ft^k_{is}$ and set $k^* := k$

    **else**

      ForwardExtension$((i,s,k), \mathcal{FQ}, \mathcal{BQ}, k^*, L)$

    **end if**

  **else**

    set $(i,s,k) := (i^b, s^b, k^b)$ and set $BQ_k := BQ_k \setminus \{(i^b, s^b, k^b)\}$

    **if** $i = O$ and $s \in F^{BA}$ **then**

      set $L^k := bt^k_{is}$ and set $k^* := k$

    **else**

      BackwardExtension$((i,s,k), \mathcal{FQ}, \mathcal{BQ}, k^*, L)$

    **end if**

  **end if**

  {queue discarding test}

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

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

  **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 $FQ_k$ and $BQ_k$ with $k \geq k^*$.

    set $K := k^* - 1$, $k^* = argmin_{k \in \{0,\ldots,K\}} L^k$

  **end if**

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

$L^0 > 1 + 1$ : no update

**Iteration 3:** select $x_2$ from $FQ_1$, forward search, $(i,k) = (x_2, 1)$

$(j,k') = (x_4, 1)$: $ft^1_4 = 6$, $FQ_1 := \{x_4\}$, 1 connection: $bt^1_4 = 1 \Rightarrow L^2 = 7$, $k^* = 2$

$(j,k') = (x_3, 2)$: $ft^2_3 = 2$, $FQ_2 := \{x_3\}$, 1 connection: $bt^0_3 = 5$, value $7 \geq L^2$ (no update)

$argmin\{ft^{k'}_{i',s'} | (i', s', k') \in \mathcal{FQ}\} = (x_3, 2)$, $ft^2_3 = 2$

$argmin\{bt^{k'}_{i',s'} | (i', s', k') \in \mathcal{BQ}\} = (x_4, 1)$ $bt^1_4 = 1$

$L^2 > 2 + 1$ : no update

**Algorithm 4** Forward multi-queue algorithm: ForwardExtension$((i,s,k),\mathcal{FQ},\mathcal{BQ},k^*,L)$

**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 $\forall k'' \leq k'$, $L^{k''} > ft^k_{i,s} + d_{ij}$ and $\forall s'' \preceq s'$, $ft^{k''}_{js''} > ft^k_{i,s} + d_{ij}$ **then**
    set $ft^{k'}_{j,s'} := ft^k_{i,s} + d_{ij}$, $p^{k'}_{j,s'} := (i,s,k)$ and $FQ_{k'} := FQ_{k'} \cup \{(j,s',k')\}$
    {Connection checking}
    **for** $(j, s^b, k'') \in \mathcal{BQ}$, $k' + k'' \leq K$, $s^b \in CS^{FA\to BA}(s')$ **do**
      **if** $L^{k'+k''} > ft^{k'}_{j,s'} + bt^{k''}_{j,s^b}$ **then**
        set $L^{k'+k''} := ft^{k'}_{j,s'} + bt^{k''}_{j,s^b}$
        **if** $k^* = -1$ or $L^{k'+k''} < L^{k^*}$ **then**
          set $k^* := k' + k''$
        **end if**
      **end if**
    **end for**
  **end if**
**end for**

**Iteration 4:** select $x_4$ from $BQ_1$, backward search, $(i,k) = (x_4, 1)$
$(j,k') = (x_2,1)$: $bt^1_2 = 6$, $BQ_1 := \{x_2\}$, 1 connection: $ft^1_2 = 1$, value $7 \geq L^2$ (no update)
$(j,k') = (x_3,2)$: $bt^2_3 = 2$, $BQ_2 := \{x_3\}$, 2 connections: $ft^0_3 = 5$, value $7 \geq L^2$ (no update)
and $ft^2_3 = 2 \Rightarrow L^4 = 4$, $k^* = 4$
$argmin\{ft^{k'}_{i',s'} | (i',s',k') \in \mathcal{FQ}\} = (x_3, 2)$, $ft^2_3 = 2$
$argmin\{bt^{k'}_{i',s'} | (i',s',k') \in \mathcal{BQ}\} = (x_3, 2)$ $bt^2_3 = 2$
$L^4 \leq 2+2$ : shortest path with 4 transfers, $K = 3$, $k^* = 2$
**Iteration 5:** select $x_3$ from $FQ_2$, forward search, $(i,k) = (x_3, 2)$
$(j,k') = (x_5,2)$: discarded since $L^2 \leq ft^2_3 + 5$, $FQ_2 := \emptyset$
$(j,k') = (x_4,3)$: $ft^3_4 = 3$, $FQ_3 := \{x_4\}$, no connection with less than 4 transfers
$argmin\{ft^{k'}_{i',s'} | (i',s',k') \in \mathcal{FQ}\} = (x_4, 3)$, $ft^3_4 = 3$
$argmin\{bt^{k'}_{i',s'} | (i',s',k') \in \mathcal{BQ}\} = (x_3, 2)$ $bt^2_3 = 2$
$L^2 > 3+2$ : no update
**Iteration 6:** select $x_3$ from $BQ_2$, backward search, $(i,k) = (x_3, 2)$
$(j,k') = (x_1,2)$: discarded since $L_2 \leq bt^2_3 + 5$, $BQ_2 := \emptyset$
$(j,k') = (x_2,3)$: $bt^3_2 = 3$, $BQ_3 := \{x_2\}$, no connection with less than 4 transfers
$argmin\{ft^{k'}_{i',s'} | (i',s',k') \in \mathcal{FQ}\} = (x_4, 3)$, $ft^3_4 = 3$
$argmin\{bt^{k'}_{i',s'} | (i',s',k') \in \mathcal{BQ}\} = (x_2, 3)$ $bt^3_2 = 3$
$L^2 > 3+3$ : no update
**Iteration 7:** select $x_4$ from $FQ_3$, forward search, $(i,k) = (x_4, 3)$
$(j,k') = (x_5,4)$: discarded since $L_4 \leq ft^+_4 1$, $FQ_3 := \emptyset$
$argmin\{ft^{k'}_{i',s'} | (i',s',k') \in \mathcal{FQ}\} = (x_3, 0)$, $ft^0_3 = 5$
$argmin\{bt^{k'}_{i',s'} | (i',s',k') \in \mathcal{BQ}\} = (x_2, 3)$ $bt^3_2 = 3$

$L^2 \leq 5+3$ : shortest path with 2 transfers, $K=1$, $k^*=0$, remove $BQ_3$
**Iteration 8:** select $x_3$ from $BQ_0$, backward search, $(i,k)=(x_3,0)$
$(j,k')=(x_1,0)$: discarded since $L_0 \leq bt_3^0+5$, $BQ_0 := \emptyset$
$(j,k')=(x_2,1)$: discarded since $bt_2^1 \leq bt_3^0+1$
$argmin\{ft_{i',s'}^{k'}|(i',s',k') \in \mathcal{FQ}\}=(x_3,0)$, $ft_3^0=5$
$argmin\{bt_{i',s'}^{k'}|(i',s',k') \in \mathcal{BQ}\}=(x_2,3)$ $bt_2^1=6$
$L^0 \leq 5+6$ : shortest path with 0 transfers, $K=0$, stop.

#### 4.5.4 Adaptation for the `BI-TD-MM-V-SPP`

A (fast) extension of bidirectional search to time-dependent shortest path (in monomodal networks) has been recently proposed by Nannicini *et al.* [17]. The concepts used in their study can be transposed without difficulty in our context.

For the forward search, we simply replace $ft_{i,s}^k + d_{ij}$ by $a_{ij}(d_{ij})$ in steps 8 and 9 of the forward extension (Algorithm 4).

For the backward search, Let $b_{ij}(t)$ denote the function giving the departure time at $i$ if arrival time at $j$ is $t$ for an arc $(i,j)$ and $t$ a possible arrival time at $j$ given by the transportation timetables. As the arrival time at $D$ is not fixed, we cannot use $b_{ij}(t)$. Instead we define the travel time $d_{ij}$ such that $t-d_{ij}$ is an upper bound of $b_{ij}(t)$. $d_{ij}$ can be simply set to the minimal duration to traverse arc $(i,j)$ given the timetable associated with $(i,j)$.

By using the timetables $a_{ij}(.)$ for the forward search and the lower bounds on the travel times $d_{ij}$ for the backward search, the time computed for a path issued from a connection is a lower bound of the actual duration. To apply the queue discarding rule, the actual travel time is computed for each encountered connection by a traversal of the backward path given the arrival time at the connection node.

## 5 Computational Experiments

### 5.1 Network and data set

The aim of these experiments is to compare TLS, MQLS and FB-MQLS for the deterministic automaton ($FB_D$-MQLS) and the non-deterministic automaton ($FB_{ND}$-MQLS), as well as to evaluate the efficiency of the proposed dominance rules on a real network. The experimental comparisons were carried out on a network covering a part of the urban area of Toulouse (France). Considered modes are bus, metro, walk and private vehicle. Viability constraints are modeled by the automaton of Figure 2. Two sets of experiments were conducted: the first one considers the transportation network as a time-independent graph and in the second one, there are time-dependent travel times for public transportation (bus and metro) but the road network remains time-independent. Table 1 details the different layers of the time-independent graph in terms of modes, nodes and arcs. Timetables for buses and metro are approximate by an average travel time for each corresponding arc in the network.

For the time-dependent network, in our example, some bus lines that have different characteristics according to the timetables are break up in several sub-modes in the multi-layer graph. For this reason, even if the time-dependent graph is the same than the time-independent graph, it has more nodes and arcs for bus and transfer modes (see table 2)

| Modes | Nodes | Arcs |
|---|---|---|
| Bus | 6 170 | 6 646 |
| Metro | 75 | 72 |
| Street | 56 774 | 146 280 |
| Transfer | - | 6 370 |
| Parking | 29 | - |
| Total | 63 048 | 159 368 |

Table 1: Time-independent network data

and it verifies the FIFO assumption.

| Modes | Nodes | Arcs |
|---|---|---|
| Bus | 9 384 | 9 047 |
| Transfer | - | 28 017 |
| Total | 65 262 | 183 416 |

Table 2: Time-dependent network data

All algorithms have been implemented in C++ and run on an 2.67 GHz Intel Xeon quad core processor W3520 with 4GB RAM under Linux Fedora 11.

### 5.2 Results

Experiments concern 100 randomly generated origin-destination pairs. All algorithms solve all problems to optimality. We obtained in average 5 non-dominated solutions and 2 transfers per itinerary. As a premiminary remark, we obtain much more efficient solutions in average than Gräbener *et al.* in [10] for the same objectives. Although networks are different we explain this difference by the fact that we do not have a dominant mode, since the private car can only be left at a limited number of nodes and we have viability constraints. The average distance is 40 km (from 30 km to 50 km), these itineraries are longer than usual itineraries in the urban part of Toulouse but it leads to a best overview of the interest of dominance rules. The start time is fixed to 08h00 am, the total number of timetable elements for the bus is 129 975 and the frequency for the metro is 3 min. In the following of this sections, all the comparisons are made in terms of CPU time but comparisons in terms of number of labels provide same conclusions and are not detailed here. All numerical results are given in the appendix.

Figure 5 shows the impact of dominance rules in terms of CPU time for the algorithms TLS, MQLS, $FB_D$-MQLS, $FB_{ND}$-MQLS in the time-independent graph. Integration of the basic dominance rules improve the results of all algorithms. The state-based dominance rule improves in turn the results of the basic dominance rules for all algorithms. Moreover, this dominance rule is more efficient for MQLS algorithms (in monodirectional or bidirectional way) than for TLS algorithm. The worst-case complexity of TLS algorithm is better than that of the MQLS algorithm (with or without the use of dominance rules). But in practice, MQLS algorithm is faster than TLS algorithm with the state-based dominance rule. The best algorithm is FB-MQLS with state-based dominance rules. For this algorithm, the best result is obtained with the deterministic backward automaton, as expected.

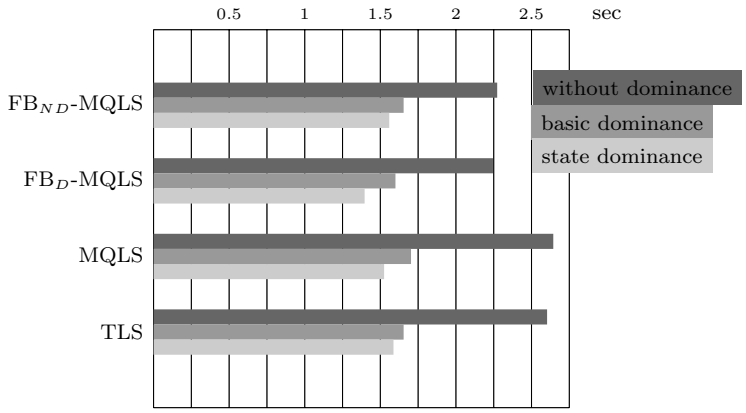



Figure 5: Impact of dominance rule in the time-independent graph

Figure 6 compares the four algorithms on the time-dependent graph, with the state-based dominance rule. Again the following rank is observed in terms of CPU time FB$_D$-MQLS<FB$_D$-MQLS<MQLS<TLS. In conclusion, both on the time-independent and on

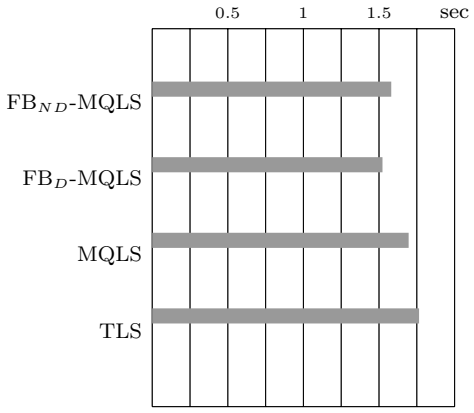



Figure 6: Comparaison of algorithms for the time-dependant graph

the time-dependent graphs, the bidirectional MQLS algorithm with state-based dominance rule and deterministic automaton for backward search is the most efficient (the CPU time is in average about 1.3 seconds in the time-independent graph and about 1.5 seconds in the time-dependent one). Another conclusion is that the use of deterministic finite state automaton in bidirectional algorithms leads to more efficient algorithms. Although these times could be certainly reduced with further acceleration techniques, our experiments allow to answer positively to the question whether the `BI(-TD)-MM-V-SPP` can be solved efficienlty on a real urban network.

# 6 On the integration of goal-oriented ($A^*$) techniques

## 6.1 Using the $A^*$ principle

The previous algorithms can be extended using the $A^*$ principle for computing point to point shortest paths. For each label $(i, s, k)$, a modified travel cost denoted by $\tilde{t}_{is}^k$ is then given by $t_{is}^k + h_i$ where $t_{is}^k$ is the travel time from the origin $O$ to $i$ and $h_i$ is an estimation

of the total travel time from $i$ to the destination $D$. $\tilde{t}_{is}^k$ represents an estimation of the shortest path form a given origin to a given destination throught node $i$ and in the algorithms the search of a label with the minimum travel time is replaced by the search of a label the minimum estimate total travel time.
We consider here that $h_i$ is independant of the states and the number of transfers and that it correspond to the euclidian distance from $i$ to $D$ over the maximal speed of the modes. Then $h_i$ is a lower bound of the real travel time from $i$ to $D$, which insure the optimality of the $A^*$ search.
Algorithms TLS and MQLS are directly extended to $A^*$ principle by using $\tilde{t}_{is}^k$ instead of $t_{is}^k$.
For the bidirectional algorithms, there are two functions of estimation: $h_i^f$ for the forward search (estimation from $i$ to $D$ in the graph) and $h_i^b$ for the backward search (estimation from $i$ to $O$ in the reverse graph) and the optimality test must be updated not to consider twice the estimations, as stated in [11].

$$ft_{i,s^f}^h + bt_{i,s^b}^q \leq max(\min_{(i',s',k')\in\mathcal{FQ}} \tilde{ft}_{i',s'}^{k'}, \min_{(i',s',k')\in\mathcal{BQ}} \tilde{bt}_{i',s'}^{k'})$$

Moreover, during the bidirectional search, some labels can be discarded with regards to previously obtained connections. Let $L^k$ denotes the value of the best obtained connection with $k$ transfers, if a label $(j, s', k')$ is obtained such that there exists a number of transfers $h$ such that $h \leq k'$ and $ft_{i,s'}^{k'} + h_i \geq L^h$ (in the forward search) then the label $(j, s', k')$ is dominated and can be discarded. The symmetric condition can also be applied in backward search.

With the conditions presented above, the $A^*$ bidirectional algorithm provides all non-dominated paths. However, the optimality condition is known to be little effective. We also consider another condition which does not guarantee any more to obtain all of the non-dominated solutions but which turns out to be more effective in practice. This condition uses twice the estimation function in the both part of the inequality:

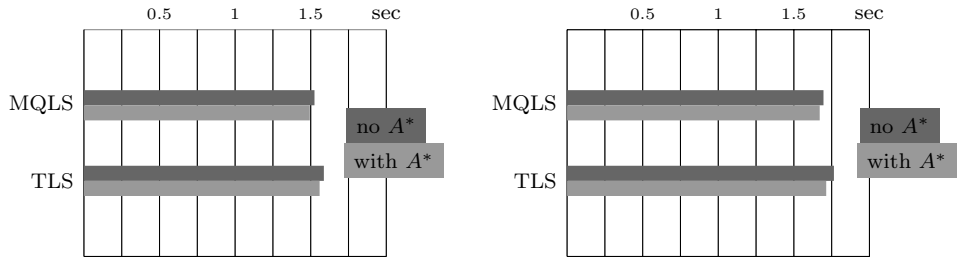
$$\tilde{ft}_{i,s^f}^h + \tilde{bt}_{i,s^b}^q \leq \min_{(i',s',k')\in\mathcal{FQ}} \tilde{ft}_{i',s'}^{k'} + \min_{(i',s',k')\in\mathcal{BQ}} \tilde{bt}_{i',s'}^{k'}$$

This condition is correct only in the cases that if the network is such that if any itinerary goes further from the origin it proportionnaly comes closer to the origin.

### 6.2 Computational experiments

Figure 7 presents the impact of $A^*$ principle for the mono-directional algorithms TLS and MQLS including the state-based dominance on the time-independent graph (Figure 8(a)) and on the time-dependant graph (Figure 8(b)). The use of $A^*$ principle improves very slightly the mono-directionnal algorithms: for TLS the improvement is about 1.76% in the time-independent graph and 2.89% in the time-dependent graph and for MQLS the impovement is about 2.03% in the time-independent graph and 1.47% in the time-dependent one.

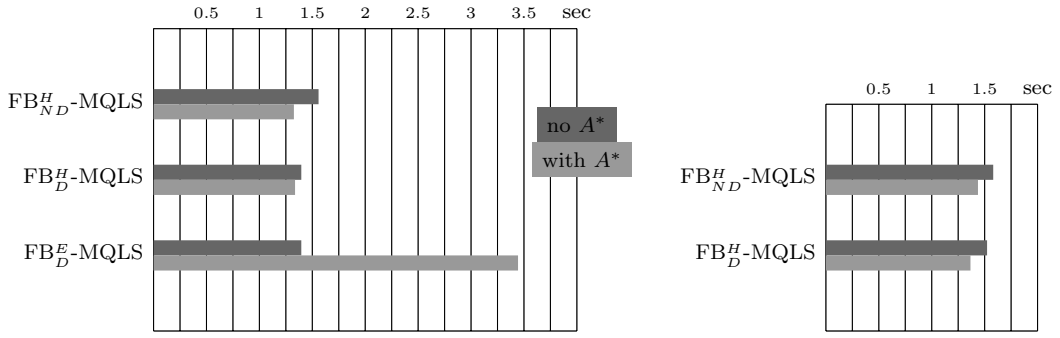
The next experiments concern the impact of $A^*$ in bidirectional algorithms (Figure 8). For that, the exact and heuristic conditions for the optimality of the connection are evaluated in the $A^*$ bidirectional algorithm with the deterministic automaton in the backward search. For the time-independent graph (Figure 8(a)), the bidirectional algorithm with $A^*$




(a) Time-independent graph (b) Time-dependant graph

Figure 7: Impact of $A^*$ for mono-directional algorithms

and the exact condition for the connection is about 2.5 times slower than the bidirectional algorithm without $A^*$. The exact condition is too weak to produce improvements in the $A^*$ bidirectional algorithms. However, the heuristic condition reduces the CPU time needed to obtain non-dominated solutions. This reduction of is about 15% for the bidirectional algorithm using a non-deterministic automaton and of about 4% for the one using deterministic automaton. The most efficient algorithm in the time-independent graph is the $A^*$ bidirectional algorithm with the heuristic condition for connection test (the CPU time is about 1.3 seconds).
The results are the same for the time-dependant graph, the exact condition is not efficient and the heuristic condition allows to obtain all the non-dominated solutions. The improvement of the $A^*$ principle is about 10.31% for the deterministic automaton and 9.11% for the non-deterministic one. Overall, even use with a heuristic component a simple implementation of $A^*$ only improves slightly the proposed algorithms, which is in line with the results obtained in [1]. Further refinements on the $A^*$ procedure, such as the design of a state-dependent estimation function could lead to better improvements.




(a) Time independent graph (b) Time-dependant graph

Figure 8: Impact of $A^*$ for bidirectional algorithms

# 7 Conclusion

We have proposed several algorithms to solve the single-source, single-destination bi-objective multimodal viable shortest path problem where path viability constraints are 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 arcs and nodes of the transportation network and in the number of states of the finite state automaton. Several improvements were brought to the topological Lozano and Storchi algorithm [13] (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 solutions 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. 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. A goal-oriented version (A*) of the three algorithms has been developped, allowing further improvements with a heuristic condition in the bidirectional algorithm.

For futher research, more experimental studies have to be carried out to evaluate the influence of the finite state automaton structure on the efficiency of the algorithms and stronger dominance rules could be exhibited, for other special cases of the state automaton. Moreover, experiments on larger transportation networks have to be realized and further acceleration techniques can be implemented. 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. The case where public transportion does not have a fixed schedule and probability distributions may be associated to arrival of passengers and transportation lines at each node is also of practical interest. Finding an "optimal" strategy for a user (that minimizes expected travel times), has been tackled via the the hypergraph model and the shortest hyperpath problem, introduced by Nguyen and Pallotino in [18] for a single public transportation mode. This approach has been extended to the bi-objective "expected travel time"/"number of transfers" multimodal viable networks in [14] but no computational experiments where reported. Adapting our algorithms to the hypergraph model is also a promising research direction.

## 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.G. Battista, M. Lucertini, and B. Simeone, Path composition and multiple choice in a bimodal transportation network. In: Proceedings of the Seventh WCTR, Sydney, 1995.

[4] 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.

[5] I. Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record*, 1645:170175, 1998.

[6] C.G. Chorus, E.J.E. Molin, T.A. Arentze, S.P. Hoogendoorn, H.J.P. Timmermans and B.V. Wee. Validation of a multimodal travel simulator with travel information provision . *Transportation Research Part C: Emerging Technologies*, 15(3):191–207, 2007.

[7] D. Delling and P. Sanders and D. Schultes and D. Wagner. Engineering Route Planning Algorithms. *Algorithmics of Large and Complex Networks, Lecture Notes in Computer Science*, 5515:117–139, 2009

[8] D. Delling and D. Wagner. Time-dependent route planning. *Robust and Online Large-Scale Optimization, Lecture Notes in Computer Science*, 5868:207–230, 2009

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

[10] T. Gräbener, A. Berro and Y.Duthen Time dependent multi-objective best path for multimodal urban routing *Electronic Notes in Discrete Mathematics*, 36:487–494, 2010.

[11] H. Kaindl and G. Kainz. Bidirectional Heuristic Search Reconsidered *Journal of Arti cial Intelligence Research*, 7:283-317, 1997.

[12] H.K. Lo, C.-W. Yip and Q.K. Wan. Modeling competitive multi-modal transit services: a nested logit approach. *Transportation Research Part C: Emerging Technologies*, 12(3-4):251–272, 2004.

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

[14] A. Lozano, G. Storchi. Shortest viable hyperpath in multimodal networks *Transportation Research Part B*, 36:853–874, 2002.

[15] E.Q.V.Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

[16] P. Modesti and A. Sciomachen A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3):495–508, 1998.

[17] 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'08), Lecture Notes in Computer Science*, vol. 5038:334–346, 2008.

[18] S. Nguyen, and S. Pallottino. Hyperpaths and shortest hyperpaths. in Combinatorial Optimization (B. Simeone, ed.) *Lecture Notes in Mathematics*, 1403, 258–271, 1989.

[19] A. Orda and R. Rom. Shortest-path and minimum delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990.

[20] 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.

[21] 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.

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

[23] 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.

# Appendix

This section details the numerical results of the computational experimentations. In the following tables:

- row *CPU Time* gives the average CPU time in milliseconds over the 100 itineraries;
- row *# Dequeued labels* represents the average number of labels dequeued of the priority queue during the search;
- row *# Enqueued labels* corresponds to the average number of labels enqueued in the priority queue during the search;
- row *# Visited labels* provides the average number of visited labels which correspond to the number of scanned successors.

Tables 3-5 correspond to the time-independent graph without $A^*$. Tables 6-8 correspond to the experiments in the time-independent graph with $A^*$. For the FB-MQLS with $A^*$, three variants are evaluated : with exact condition and deterministic automaton in the backward search, with heuristic condition and deterministic automaton in the backward search, and with heuristic condition and non-deterministic automaton in the backward search. Last, Tables 9-10 give the results obtained by algorithms TLS, MQLS and FB-MQLS with or without the $A^*$ principle in the time-dependent graph.

| | TLS | MQLS | $FB_D$-MQLS | $FB_{ND}$-MQLS |
|---|---|---|---|---|
| CPU Time | 2 603 | 2 644 | 2 245 | 2 273 |
| # Dequeued labels | 555 900 | 555 899 | 386 391 | 409 516 |
| # Enqueued labels | 629 159 | 61 0193 | 430 994 | 456 657 |
| # Visited labels | 1 428 760 | 1 428 760 | 1 016 180 | 1 076 450 |

Table 3: Comparison of the proposed algorithms without dominance rule in the time-independent graph

| | TLS | MQLS | $FB_D$-MQLS | $FB_{ND}$-MQLS |
|---|---|---|---|---|
| CPU Time | 1 654 | 1 703 | 1 600 | 1 654 |
| # Dequeued labels | 376 339 | 377 119 | 283 721 | 306 942 |
| # Enqueued labels | 420 374 | 411 903 | 313 798 | 339 536 |
| # Visited labels | 960 483 | 962 706 | 738 354 | 798 842 |

Table 4: Comparison of the proposed algorithms with the basic dominance rule in the time-independent graph

| | TLS | MQLS | $FB_D$-MQLS | $FB_{ND}$-MQLS |
|---|---|---|---|---|
| CPU Time | 1 587 | 1 525 | 1 396 | 1 559 |
| # Dequeued labels | 357 838 | 337 687 | 255 464 | 278 561 |
| # Enqueued labels | 399 459 | 368 021 | 281 770 | 307 381 |
| # Visited labels | 912 819 | 857 944 | 660 971 | 721 159 |

Table 5: Comparison of the proposed algorithms with the state-based dominance rule in the time-independent graph

| | TLS | MQLS | $FB^H_D$-MQLS | $FB^H_{ND}$-MQLS | $FB^E_D$-MQLS |
|---|---|---|---|---|---|
| CPU Time | 2 496 | 2 560 | 1 944 | 2 026 | 5 320 |
| # Dequeued labels | 541 619 | 541 618 | 338 490 | 366 260 | 738 746 |
| # Enqueued labels | 614 485 | 595 728 | 379 994 | 415 358 | 815 574 |
| # Visited labels | 1394 650 | 1 394 650 | 894 320 | 966 229 | 1 908 960 |

Table 6: Comparison of the proposed $A^*$ algorithms without dominance rule in the time-independent graph

| | TLS | MQLS | $FB^H_D$-MQLS | $FB^H_{ND}$-MQLS | $FB^E_D$-MQLS |
|---|---|---|---|---|---|
| CPU Time | 1 659 | 1 693 | 1 455 | 1 492 | 5 320 |
| # Dequeued labels | 369 146 | 370 041 | 259 671 | 276 089 | 738 746 |
| # Enqueued labels | 413 389 | 405 046 | 288 897 | 310 959 | 815 574 |
| # Visited labels | 943 528 | 946 037 | 678 385 | 719 805 | 1 908 960 |

Table 7: Comparison of the proposed $A^*$ algorithms with the basic dominance rule in the time-independent graph

| | TLS | MQLS | $FB^H_D$-MQLS | $FB^H_{ND}$-MQLS | $FB^E_D$-MQLS |
|---|---|---|---|---|---|
| CPU Time | 1 559 | 1 494 | 1 337 | 1 326 | 3 444 |
| # Dequeued labels | 351 998 | 332 027 | 235 079 | 251 082 | 51 5076 |
| # Enqueued labels | 393 947 | 362 668 | 260 762 | 282 515 | 566 297 |
| # Visited labels | 899 175 | 844 700 | 610 145 | 650 729 | 1 321 060 |

Table 8: Comparison of the proposed $A^*$ algorithms with the state-based dominance rule in the time-independent graph

| | TLS | MQLS | $FB_D$-MQLS | $FB_{ND}$-MQLS |
|---|---|---|---|---|
| CPU Time | 1 765 | 1 696 | 1 523 | 1 581 |
| # Dequeued labels | 344 125 | 345 976 | 248 035 | 271 532 |
| # Enqueued labels | 386 538 | 378 042 | 275 912 | 301 892 |
| # Visited labels | 911 676 | 916 213 | 672 985 | 737 073 |

Table 9: Comparison of the proposed algorithms with the state-based dominance rule in the time-dependent graph

| | TLS | MQLS | $FB_D^H$-MQLS | $FB_D^H$-MQLS |
|---|---|---|---|---|
| CPU Time | 1 714 | 1 671 | 1 366 | 1 437 |
| # Dequeued labels | 339 288 | 341 616 | 233 812 | 237 393 |
| # Enqueued labels | 382 254 | 374 116 | 261 236 | 269 807 |
| # Visited labels | 900 099 | 905 769 | 635 906 | 643 260 |

Table 10: Comparison of the proposed $A^*$ algorithms with the state-based dominance rule in the time-dependent graph