HAL
open science

State-based accelerations and bidirectional search for bi-objective multimodal shortest paths

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

To cite this version:

Christian Artigues, Marie-José Huguet, Fallou Gueye, Frédéric Schettini, Laurent Dezou. State-based accelerations and bidirectional search for bi-objective multimodal shortest paths. Transportation research. Part C, Emerging technologies, 2013, 27, pp.233-259. 10.1016/j.trc.2012.08.003 . hal-00624984v2

HAL Id: hal-00624984

https://hal.science/hal-00624984v2

Submitted on 28 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

State-based accelerations and bidirectional search for bi-objective multi-modal shortest paths

Christian Artigues[1,2], **Marie-José Huguet**[1,3],
Fallou Gueye[1,3,4], **Frédéric Schettini**[4] **and Laurent Dezou**[4]

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France
[3] Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[4] MobiGIS; ZAC Proxima, rue de Lannoux, 31310 Grenade Cedex France

artigues@laas.fr, huguet@laas.fr

Abstract

Taking into account the multi-modality of urban transportation networks for computing the itinerary of an individual passenger introduces a number of additional constraints such as mode restrictions and various objective functions. In this paper, constraints on modes are gathered under the concept of viable path, modeled by a non deterministic finite state automaton (NFA). The goal is to find the non-dominated viable shortest paths considering the minimization of the travel time and of the number of modal transfers. We show that the problem, initially considered by Lozano and Storchi [15], is a polynomially-solvable bi-objective variant of the mono-objective regular language-constrained shortest path problem [2, 8].

We propose several label setting algorithms for solving the problem: a topological label-setting algorithm improving on algorithms proposed by Pallottino and Scutellà [23] and Lozano and Storchi [15], a multi-label algorithm using buckets and its bidirectional variant, as well as dedicated goal oriented techniques. Furthermore, we propose a new NFA state-based dominance rule. The computational experiments, carried-out on a realistic urban network, show that the state-based dominance rule associated with bidirectional search yields significant average speed-ups. On an expanded graph comprising 1 859 350 nodes, we obtain on average 3.5 non-dominated shortest paths in less than 180 ms.

keywords: bi-objective regular language-constrained shortest path problem, multi-modal transportation, finite state automaton, label-setting algorithms, state-based dominance rule, bidirectional search, state-based estimated travel times.

1 Introduction

Computing shortest paths in the context of mono-modal passenger transportation, where a single transportation mode (e.g. private vehicle, bus, subway) is used during the passenger's itinerary, has been the subject to extensive research since the publication of Dijkstra's algorithm in the 1950s. Among the considered extensions of the basic shortest path problem, 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 fast in large-scale network, either in a time-independent or in a time-dependent context, is not a challenge anymore for labeling algorithms [6, 7, 19].

The case of multi-modal passenger transportation, which lies in the transportation of one or several passengers with different modes during the same itinerary, has been much less addressed. However, multi-modal transportation is subject to a growing interest in the research community, as multi-modality is now widely accepted for urban transportation as a necessary alternative to the exclusive use of private vehicles.

In this paper, we consider a bi-objective shortest path problem in a multi-modal urban transportation network: the minimum travel time/minimum number of transfers regular language-constrained shortest-path problem, initially considered by Lozano and Storchi [15], and denoted by BI-REGL-CSPP by analogy to its mono-objective variant: the regular language (or label)-constrained shortest path problem REGL-CSPP defined by Barett [2] and recently considered in [1, 8, 22][1].

The problem can be briefly stated as follows. Consider a network $G(V, E)$. Each node $i \in V$ is "colored" by a mode m_i (e.g. bus, walk, car, subway). Each arc $(i, j) \in E$ has a weight corresponding to a travel time in mode m if $m_i = m_j = m$ or to a transfer time if $m_i \neq m_j$. Furthermore, there are so-called path viability constraints. More precisely, the sequence of modes induced by a path is constrained in such a way that the string obtained by concatenating the path modes (considered as elements of an alphabet) must belong to a formal language described by a non-deterministic finite state automaton (NFA). Considering an origin node O and a destination node D, the goal is to find an $O - D$ path for each of the non-dominated points in the space of two min sum objectives: the minimum total travel time and the minimum number of transfers (mode changes along the path). As in [15], we restrict ourselves to a time-independent version of the problem (travel times are constant) but we will briefly describe an extension scheme of the proposed algorithms to FIFO networks in Section 7.

Lozano and Storchi [15] implicitely assume (without mentioning it) that the problem can be solved in polynomial or pseudo-polynomial time. They proposed a label correcting algorithm to solve the problem, based on a topological labeling algorithm proposed by Pallottino and Scutellà in [23].

In this paper, we first prove the problem can actually be solved in polynomial time. We propose a label setting extension of the Pallottino and Scutellà algorithm and several improvements to the Lozano and Storchi approach, among which a new state-based dominance rule. We also propose a multi-label approach based on buckets, from which we derive a bidirectional label setting algorithm. Goal-oriented techniques with state-dependent estimated travel times are finally presented. We consider an application to the urban area of Toulouse (France), the purpose of which being to evaluate the tractability of the considered problem on a real-life network, as no computational experiments were reported in [15]. On this practical network, we compare the different algorithms. In particular, we study the impact of the dominance rules and of the size and nature of the NFA on the CPU time and on the number of touched nodes.

Section 2 briefly recalls the existing literature on the multi-modal shortest path problem, so as to introduce our motivations for the present study. Section 3 formally defines the problem and Section 4 establishes its complexity. The proposed algorithms are pre-

[1] Lozano and Storchi [15] referred to the problem as the multi-modal viable shortest-path problem

sented in Section 5. Their performance on the considered real network are compared in Section 6. Further extensions and concluding remarks are presented in Section 7.

2 Literature review

Multi-modal transportation raises network modeling issues [5, 14]. A simple way of modeling the network, used by many authors [18, 15, 4], lies in assuming that the set of nodes is partitioned according to the modes. An arc linking two nodes of different 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 sequences 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 [18], as an 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, (possibly infinite) mode-dependent travel times were used by Ziliaskopoulos and Wardell [27], together with an arc representation, allowing to design mode constraints involving three nodes. A more general way of modeling the multi-modal constraints (among other applications) was proposed by Barett *et al.* [2]. Each mode being viewed as an element of an alphabet and each arc of the network being labeled by a mode, the mode restrictions can be described by a regular language over the alphabet. The multi-modal shortest path problem then amounts to a regular language-constrained shortest path problem (RegL-CSPP). 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, 8, 22, 26, 25], practical implementation issues of this method have been discussed. Barett *et al.* [1] proposed $A*$ and bidirectional accelerations while Delling *et al.* [8] and Pajor [22] proposed, in addition, core-based and access node-based speed-up techniques and also considered time-dependent networks. With these techniques, they obtain a CPU time of 2.3 milliseconds on average for computing the $O-D$ shortest path on a large intercontinental network with 50 700 647 nodes, 125 939 503 edges and a 3 state-automaton. Considering deterministic finite-state automaton (DFA) as input, Sherali *et al.* [25] extended the problem to time-dependence and propose a strongly polynomial algorithm for FIFO graphs. Sherali and Jeenanunta [26] further extended 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 RegL-CSPP is 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. To that purpose, two general classes of multi-objective multi-modal shortest path problems have already been considered in the literature: namely, (pseudo-)polynomial problems and NP-hard problems.

In [23], Pallottino and Scutellà considered the Bi-RegL-CSPP problem without the viability constraints (any $O-D$ path is feasible) but with the constraint that the number of transfers does not exceed a maximum number $k_{\max}$. Although the general bi-objective shortest path problem with min-sum objective is NP-hard, they showed that, in this case, the problem can be solved in pseudo-polynomial time (in n, the number of nodes in the network, and $k_{\max}$), as it belongs to the category of bi-objective shortest path problems

for which one of the two objectives takes its values in a discrete and finite set. They proposed a topological labeling approach where a label (i, k) represents an $O-i$ path with k transfers.

In [15], Lozano and Storchi directly extended this problem and the topological algorithm, so as to integrate mode restrictions using a NFA in a time-independent network, defining the Bi-RegL-CSPP considered in the present paper. Note they performed this extension without formally proving that the problem is (pseudo-)polynomial. We will prove in the sequel that the problem can actually be solved in polynomial time in n and $|S|$, the number of states of the NFA.

Bielli *et al* [4] considered a simplified version of the NFA model but include time-dependent arcs and time penalties for turning movements. 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 [23] algorithm, with labels on arcs. Experimentations 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 Bi-RegL-CSPP. Besides providing new algorithms and acceleration techniques, our main purpose in this paper is consequently to study the tractability of solving this bi-objective problems on a real network in reasonable computational time.

A more general class of approaches considers several different objective functions and, mostly, proposes extensions to multi-modality of the (NP-hard) bi-objective shortest path problem. Although we focus in this paper on the polynomial Bi-RegL-CSPP, we mention the recent experimental study carried out by Gräbener *et al.* [10] for a more general problem since the number of transfers and minimum time objectives were also considered among other objectives in their study. They present an extension of Martin's algorithm [17] to deal with the multi-modal, multiobjective shortest path problem, 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 paths 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-objective multi-modal problem 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-RegL-CSPP 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. Furthermore, we evaluate the efficiency of a new NFA state-based dominance rule and of a new bidirectional label-setting algorithm based on buckets.

3 Problem statement

3.1 Definition

Let M denotes the set of modes. The multi-modal transportation network is modeled by a multi-layered network $G(V, E)$ with $|V| = n$ such that each layer corresponds to a mode $m \in M$. Hence, a mode $m_i \in M$ is defined for each node $i \in V$ while 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.

In terms of multi-modal characteristics, each path in G yields a sequence (or string) of modes. Among all strings of modes, only some subsets of strings are acceptable according to the feasibility constraints (or preferences). The acceptable mode sequences are represented via a non-deterministic finite state automaton (NFA), possibly issued from a user-defined regular expression [2]. This NFA 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 $\delta : M \times M \times S \to 2^S$ is the transition function such that $\delta(m, m', s)$ gives the set of states 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 infeasible. Note that the case were $\delta(m, m', s)$ is either the empty set or a singleton yields a deterministic finite state automaton (DFA).

A viable path is a path in G from the origin node O to the destination node D satisfying the constraints represented by the NFA. A path is viable if it starts with O (in state s_0) and reaches D in a final state $s \in F$.

We consider both the "minimum time" and "minimum number of transfers" objectives. We first recall definitions on multi-objective 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 NFA scenarios

For our experimental evaluation on a real network, we consider 4 modes $\{w, b, c, s\}$ (walking, bus, private car, subway) and 3 different mode configuration scenarios: $M_1 = \{w, b\}$ (walk and bus only), $M_2 = \{w, b, s\}$ (walk, bus and subway only) and $M_3 = \{w, b, c, s\}$ (all modes).

For mode scenario M_3 (inspired by the one provided in [15]), we consider the following viability constraints. The private car can be taken only from O and, once left, cannot be taken again. For the subway, we assume that it can be taken at any time except at the origin but, once left, cannot be taken again, which corresponds to a user preference. In this case study we consider also that the destination is not in the subway and cannot be reached by car.

These viability constraints can be modeled by a NFA with $|S| = 5$ represented in Figure 1. Transition arcs between states are labeled by a mode $m \in M$ where $M = \{w, b, c, s\} \cup \{m_O\}$, m_O being a fictitious 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 $s' \in \delta(m_i, m_j, s)$ 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.

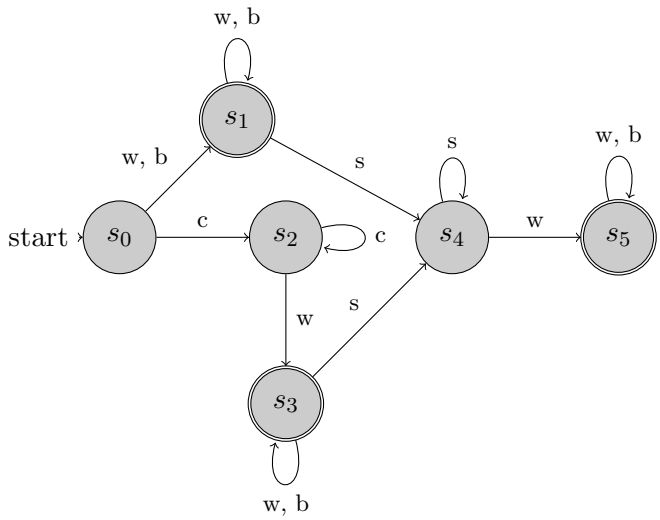


Figure 1: Initial NFA for mode scenario $M_3 = \{w, b, c, s\}$

The state at origin is denoted s_0 and other states have the following meaning:

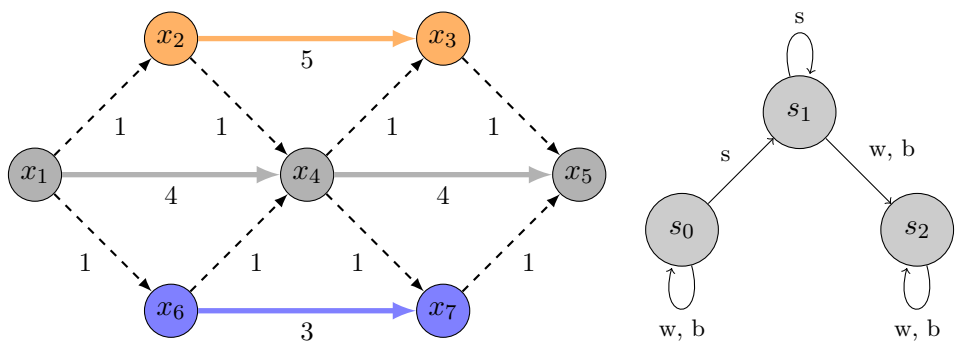
- s_1: the private car was not taken at the origin O and, so, mode c is forbidden for the remaining of the travel, while subway has not been taken yet;
- s_2: the private car was taken at the origin O and has not been left yet;
- s_3 the private car cannot be taken anymore since it has already been taken and left while subway mode has not been taken yet;
- s_4: the subway has been taken but not left;
- s_5: the subway has been left.

We consider that 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 subway, so she/he must leave the subway 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.
For mode scenario M_2, as the car is not considered anymore, states s_2 and s_3 are removed and only 3 states remain in the NFA (see Figure 2(b)). And, for subway mode, we consider the same viability constraints as in mode scenario M_3.
For mode scenario M_1, as there are no constrained mode anymore, the NFA is empty. Note that all considered input NFAs are also DFAs.

3.3 Bi-RegL-CSPP example

Figure 2(a) shows an (unrealistic) multi-modal network with 7 nodes corresponding to mode scenario $M_2 = \{w, b, s\}$. Nodes x_2 and x_3 are bus (b) nodes; nodes x_1, x_4 and x_5

are walk (w) nodes; nodes x_6 and x_7 are subway (s) modes. Dashed arcs correspond to transfer arcs. The NFA corresponding to M_2 is displayed in Figure 2(b).



(a) multi-modal network

(b) NFA for mode scenario $M_2 = \{w, b, s\}$

Figure 2: a Bi-RegL-CSPP with 7 nodes, 12 arcs, 3 modes and a 3 state-NFA

With this NFA, there are three non-dominated solutions to the Bi-RegL-CSPP from x_1 to x_5: path (x_1, x_4, x_5) with 0 transfer and a travel time equals to 8, path (x_1, x_6, x_7, x_5) with 2 transfers and a travel time of 5 and path $(x_1, x_2, x_4, x_3, x_5)$ with 4 transfers and a travel time equals to 4. Note that path $(x_1, x_6, x_4, x_7, x_5)$ of cost 4 is infeasible as the automaton, initially in state s_0, is set to s_1 taking transfer arc (x_1, x_6) and then to s_2 through transfer arc (x_6, x_4) from which it is not possible to take the subway anymore.

4 Complexity

In this section we establish the complexity status of the Bi-RegL-CSPP which was not stated in [15] although it was implicitly considered that the problem is pseudo-polynomial in n and $k_{\max}$ (the maximum number of transfers). This seems intuitively correct. Indeed, Barett [2] proved that the mono-objective RegL-CSPP is polynomially solvable, while Pallottino and Scutellà showed that the bi-objective minimum time/minimum number of transfers multi-modal shortest path problem (without viability constraints) is pseudo-polynomially solvable under a threshold $k_{\max}$ on the number of transfers.

Theorem 4.1. *The* Bi-RegL-CSPP *can be solved in a time polynomial in* $n|S|$.

Proof. Suppose in a first step that the number of transfers must not exceed a threshold $k_{\max}$. We build an expanded graph by creating a node per tuple (i, s, k) for $i \in V$, $s \in S$, $k \in \{0, 1, \ldots, k_{\max}\}$ and an arc from node (i, s, k) to (j, s', k') valuated by d_{ij} where $(i, j) \in E$, $s' \in \delta(m_i, m_j, s)$ and $k' = k$ if $m_i = m_j$ and $k' = k + 1$ if $m_i \neq m_j$. By solving the single-source, multi-destination mono-objective shortest path problem in this graph from node $(0, s_0, 0)$ (e.g. with Dijkstra's algorithm) we obtain all non-dominated solutions, simply by scanning the shortest paths ends (D, s, k) for $s \in F$ and $k \in \{0, 1, \ldots, k_{\max}\}$. This statement is a straightforward extension of the proof proposed by Barett [2] for the RegL-CSPP.

As the number of nodes in the expanded graph is equal to $n|S|k_{\max}$, we just showed that the problem is pseudo-polynomially solvable. Consider now a non-dominated $O - D$ path in $G(V, E)$. We claim that the number of times this path visits a given node i is not

larger than $|S|$, the number of states of the NFA. Indeed, suppose the path visits twice a node i in state s, i.e from node (i, s, k) to node (i, s, k') of the expanded graph. Then, the subpath $(i, s, k), \ldots, (i, s, k')$ can obviously be removed without increasing the number of transfers and the travel time nor violating the viability constraints, which contradicts the assumption that the path is non-dominated. It follows that $k_{\max}$ cannot be larger than $n|S|$ and so the above-described algorithm is polynomial in n and $|S|$. □

Note that setting $k_{\max}$ to a number lower than $n|S|$ accelerates the search and corresponds to a reasonable restriction in most applications.

5 Algorithms

In this section, we propose several algorithms and a new state-based dominance rule to solve the Bi-RegL-CSPP. All algorithms are based on a label setting principle which is described in Section 5.1. The dominance rules are presented in Section 5.2. The first algorithm (TLS), a label-setting version of the topological label algorithm proposed by [15] with additional corrections and improvements, is given in Section 5.3. The second algorithm (MQLS), described in Section 5.4, is a new label setting algorithm based on buckets. Section 5.5 presents the third algorithm (FB-MQLS), a bidirectional (Forward-Backward) adaptation of MQLS. Finally, Section 5.6 presents state-based goal oriented (A*) techniques for the unidirectional algorithms.

5.1 Label setting principle

The proposed algorithms use labels to represent paths. Let (i, s, k) denote a label representing a path from the origin to node i in state s and using k transfers. Each label has two attributes: t^k_{is} which denotes the arrival time on i and p^k_{is} which denotes the predecessor label of (i, s, k) on the path. Note that no algorithm needs to store more than one label (i, s, k) for given i, s and k.

All the proposed algorithms implement differently the following basic principles. Initially, a label $(O, s_0, 0)$ is generated with $t^0_{Os_0} = 0$ and $p^0_{Os_0} = (O, s_0, 0)$. 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^k_{is} is removed from Q and marked-up, as t^k_{is} is the shortest time from O to i in state s with k transfers. Let $Mark^k_{is} \in \{true, false\}$ denote the mark indicator and let $\mathcal{M}$ denote the set of marked nodes. Then, the direct successors of node i are scanned. For each successor j, viability of the arc (i, j) is checked according to multi-modal restrictions and obtained labels (j, s', k') are considered for all $s' \in \delta(m_i, m_j, s)$, $k' = k$ if $m_i = m_j$ or $k' = k + 1$ if $m_i \neq m_j$. If one of the three following conditions holds,

(i) label (j, s', k') was never visited;

(ii) the Bellman condition does not hold ($t^{k'}_{js'} > t^k_{is} + d_{ij}$);

(iii) dominance rules do not apply (see Section 5.2),

the time and predecessor of (j, s', k') are updated with $t^{k'}_{js'} \leftarrow t^k_{is} + d_{ij}$ and $p^{k'}_{js'} \leftarrow (i, s, k)$ and the label is inserted in Q or, if it is already present, Q is updated according to $t^{k'}_{js'}$ decrease. Otherwise the label is discarded.

5.2 Dominance rules and state reduction

In this section, we give dominance rules allowing to discard labels. During the extension process, a label can be discarded if it can be proven that it cannot be extended to a better solution than another already generated label .

A first dominance rule, the basic dominance rule, is linked to the bi-objective optimization. It was used by Pallottino and Scutellà in [23] to solve the bi-objective multi-modal shortest path without viability constraints.

Proposition 5.1 (Basic dominance rule)**.** *Consider two distinct labels (i, s, k) and (i, s, k'). If $k \leq k'$ and $t_{is}^{k} \leq t_{is}^{k'}$, label (i, s, k') can be discarded (because it is dominated by the label (i, s, k)).*

Enforcing this dominance rule yields only different solutions: if there are two solutions with same travel times and same numbers of transfers, only one of them is kept. This first dominance rule compares, for a given node, only labels having the same state.

The second dominance rule allows to go further by comparing, for the same node i, labels having different states.

For that, 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,

Definition 5.1. *State s dominates state s' ($s \preceq s'$) if any input mode string accepted by s' is also accepted by s.*

Lozano and Storchi [15] proposed a state-based dominance rule (Preference theorem) applying to the NFA they consider. In Appendix A, we underline a few limitations of this theorem and we opt for new a state-based different dominance rule. We define binary relation $\preceq$ as follows.

Definition 5.2. *$s \preceq s'$ if for any mode pair $(m, m') \in M$, such that m is a feasible mode for state s, one of the following conditions holds:*

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

Note $\preceq$ is reflexive ($s \preceq s$) and transitive, so $\preceq$ is a preorder on the set of states.

Example. In the NFA presented in Figure 1, we have $s_1 \preceq s_5$. Indeed, when $m \in \{c, s\}$, as it is infeasible to be in s_1 (or in s_5) with $m = c$ or $m = s$, we have: $\delta(m, m', s_1) = \emptyset$ and $\delta(m, m', s_5) = \emptyset$, $\forall m' \in M$. When $m \in \{w, b\}$, we can have the following transitions:
$\delta(m, w, s_1) = s_1$ and $\delta(m, w, s_5) = s_5$
$\delta(m, b, s_1) = s_1$ and $\delta(m, b, s_5) = s_5$
$\delta(m, s, s_1) = s_4$ and $\delta(m, s, s_5) = \emptyset$
$\delta(m, c, s_1) = \emptyset$ and $\delta(m, c, s_5) = \emptyset$ (ie. the car mode cannot be used)

As a preliminary remark, we can use preorder $\preceq$ to reduce the number of states of the automaton.

Proposition 5.2. *If $s \preceq s'$ and $s' \preceq s$, s and s' can be merged into a single state without modifying the viability constraints.*

Proof. This condition is just a particular case of standard NFA reduction rules [13]. □

This state merging condition can be used to reduce the initial automaton presented in Figure 1. After the reduction $\preceq$ is also antisymmetric and defines a partial order on the set of states. In this figure, states s_1 and s_3 verify the above-described condition and we can build the reduced automaton of Figure 3.

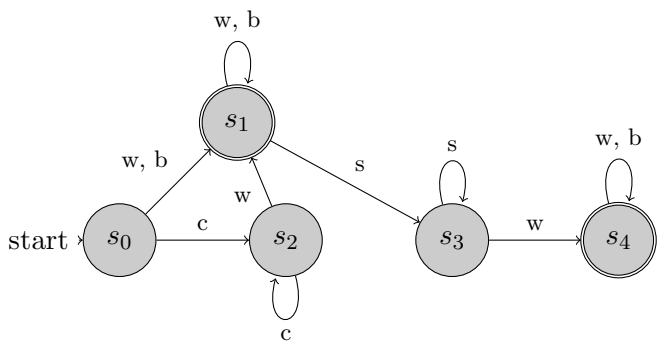


Figure 3: Reduced NFA for mode scenario $M_3 = \{w, b, c, s\}$

Such a reduction is beneficial for the computational requirements as the previous section and computational experiments show that the complexities of the presented algorithms depend on the number of states of the NFA. We now propose the following state-based dominance rule.

Theorem 5.1 (State-based dominance rule)**.** *Consider two labels (i, s, k) and (i, s', k') with $k \leq k'$, $s \preceq s'$ and $t_{is}^{k} \leq t_{is'}^{k'}$. Label (i, s, k) dominates (i, s', k'), which can be discarded.*

Proof. From the definition of relation $\preceq$, we know that any transition from s' yields the same state as any transition from s. So label (i, s, k) has at least the same extension possibilities as label (i, s', k'). The remaining conditions ($k \leq k'$ and $t_{is}^{k} \leq t_{is'}^{k'}$) show that any extension of (i, s', k') is dominated by an extension of (i, s, k). □

In the reduced automaton for mode scenario $M_3 = \{w, b, c, s\}$ (Figure 3), we see that $s_1 \preceq s4$ is the only dominance relation. The automaton for mode scenario $M_2 = \{w, b, s\}$ of Figure 2(b) cannot be reduced and dominance relation $s_0 \preceq s_2$ holds.

Note that if transitions are represented by a matrix, all dominance relations can be established during preprocessing in $O(|M|^2|S|^2)$ time.

In the algorithms described in the subsequent sections, different definitions of set $\mathcal{D}_s$ can be used to parametrize the use of dominance rules:

- $\mathcal{D}_s = \emptyset$ corresponds to no dominance checking (except the Bellman condition).
- $\mathcal{D}_s = \{s\}$ corresponds to basic dominance checking.
- $\mathcal{D}_s = \{s' \in S | s' \preceq s\}$ corresponds to state-based dominance checking.

5.3 Topological label-setting (TLS) algorithm

5.3.1 Description

The topological Pallottino and Scutellà [23] algorithm was extended by Lozano and Storchi algorithm [15] to path viability modeled by a NFA. We describe below a label-setting variant named TLS (the original algorithm of [15] being described as a label-correcting algorithm) with additional improvements and corrections.

This algorithm (pseudo-code is given in Algorithm 1 and label setting procedure is given in Algorithm 2 of Appendix B) iterates on the number of transfers from 0 to $k_{\max}$. At each iteration k, it searches for the shortest path with exactly k transfers. To that purpose, data structure Q, used for storing labels, is made of two priority queues Q^{now} and Q^{next}, where Q^{now} contains labels with k transfers and Q^{next} contains labels with $k+1$ transfers. Initially, Q^{now} contains only 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} and the label extension principle is applied with some modifications regarding the general label setting algorithm. To check the dominance rules, a variable, denoted by $BestValue_{j,s'}$, stores all algong the search the shortest travel time found so far to reach node j in state s' (i.e. with at most k transfers). The variable is used to have an $O(1)$ basic dominance checking (instead of enumerating all $t_{js'}^{k''}$ such that $k'' < k'$) and a $O(|\mathcal{D}_{s'}|)$ state-based dominance checking (see Section 5.3.2). If the dominance condition holds, the label is discarded. Otherwise, if $k = k'$, $BestValue_{j,s'}$ is updated and the new label (j, s', k') is inserted in Q^{now}. If $k' = k+1$, the label is inserted in Q^{next}.

Back to the TLS main procedure (Algorithm 1), as soon as the destination D (in a final state) is dequeued from Q_{now} or if 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 the maximum number of transfers $k_{\max}$ is reached.

In addition, the proposed version of TLS algorithm presents some improvements to the Lozano and Storchi algorithm:

- A label-setting algorithm is proposed in place a label-correcting algorithm, following the recommendations of [26].
- At each iteration, when travel time of a new label is computed, the best travel time of previous solutions, denoted by $BestLastSol$ is also used to prune labels (in addition to $BestValue$).
- In [15], a single label t_{is} is used per node-state pair (i, s) for both extensions in Q^{now} with k transfers and in Q^{next} with $k+1$ transfers. However, a path with k transfers may not be extended because it has a longer time than a path with $k+1$ transfers. More details are given on this issue in Appendix A. We correct this problem by defining two labels t_{is}^{k} and t_{iq}^{k+1} (see Appendix A for a more detailed explanation).
- We use the state-based dominance rule proposed in Section 5.2 instead of the one proposed in [15] (see Appendix A for justifications).

With TLS, solutions are obtained in increasing numbers of transfers and decreasing travel times. In Appendix C, the algorithm is applied to the instance described in Figure 2 with the state-based dominance rule. 12 labels are marked and 18 labels are reached.

5.3.2 Complexity

Complexity of dominance rule checks.
The basic dominance rule on label (j, s', k') can be performed in $O(1)$. Indeed, 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 $BestValue_{js'}$. A label (j, s', k') is dominated if $t^{k'}_{js'} \geq BestValue_{js'}$, 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 $BestValue_{j,s''} \leq t^k_{is} + d_{ij}$ for all states s'' such that $s'' \preceq s'$.

Complexity of TLS 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 $n|S|$. 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 $O(k_{max}|S||E|(|S| + \log(n|S|)))$.

5.4 Multi-queue label-setting (MQLS) algorithm

5.4.1 Description

We propose an alternative algorithm that computes the shortest paths in increasing order of the time criterion values and 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 buckets (priority queues) such that each bucket $Q_k \in \mathcal{Q}$ 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 number of buckets 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 $Q_{k'}$ by the label extension procedure. The algorithm stops when the shortest path with 0 transfer is found or when all queues are empty. The algorithm pseudo code is given in Algorithm 3 and the label extension procedure is given in Algorithm 4 of Appendix B. In Appendix C, the algorithm is also applied to the instance described in Figure 2. 12 labels are marked and 16 labels are reached by the search.

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 5.3. *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.

5.4.2 Complexity

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) has 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

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

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

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

The worst-case time complexity is increased compared to the TLS algorithm by a $k_{\max}$ factor.

5.5 Bidirectional Multi-Queue Label Setting Algorithm (FB-MQLS)

We propose a bidirectional adaptation of MQLS, taking advantage of the multi-queue characteristics. There are two main issues in designing a bidirectional algorithm for the considered multi-modal problem. The first issue consists in modeling backward path viability. The second issue lies in exploiting the connection between a forward and backward label in the bi-objective context. The multi-queue structure will be here fully exploited as several label queues will be discarded when a given connection condition is reached. These issues are addressed in Section 5.5.1 and the algorithm FB-MQLS is described in Section 5.5.2.

5.5.1 General Principles

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 $ft^k_{i,s}$ representing paths reaching i in state s with k transfers and BQ_k contains backward labels $bt^k_{i,s}$ representing paths originating from i with k transfers in state s.

Modeling backward path viability

We exhibit below three different possibilities to model backward path viability. Let

$A^f = (S^f, M, \delta^f, s_0^f, F^f)$ denotes the automaton for the forward search and $A^b = (S^b, M, \delta^b, s_D^b, F^b)$ the automaton for the backward search. To obtain A^b from A^f, the first possibility is simply to reverse the arcs of A^f as done in [22].

In our case, for mode scenario M_3, although the forward automaton is a DFA, the obtained state automaton becomes non-deterministic (see Figure 4(b)). In this figure, the initial state (at destination) is $s_D^{'b}$. Final states are $s_1^{'b}$ (departure by walk or bus) and $s_2^{'b}$ (departure by private car).

Transition function $\delta^b(m_i, m_j, s)$ gives a set of possible states.

For example $\delta^b(m_D, w, s_D) = \{s_1^{'b}, s_4^{'b}\}$ (where m_D denotes the mode at the destination). This means what when arriving by walk at the destination, it could be that the subway was taken (state $s_4^{'b}$) or was not taken (state $s_1^{'b}$).

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 it can be that for a given non-deterministic automaton with $|S|$ states, the equivalent deterministic automaton has about $2^{|S|}$ states. An issue then, is to generate the deterministic automaton with a minimal number of states. Note that this issue also applies to the forward automaton which can be non-deterministic if it is obtained from a regular expression.

A third possibility is to obtain a (ϵ-free) NFA through the reverse regular expression, which can be done in polynomial time [24]. For example, the forward regular expression corresponds to $c^*(w,b)^*s^*(w,b)^*$ and the reverse regular expression corresponds to $(w,b)^*s^*(w,b)^*c^*$.

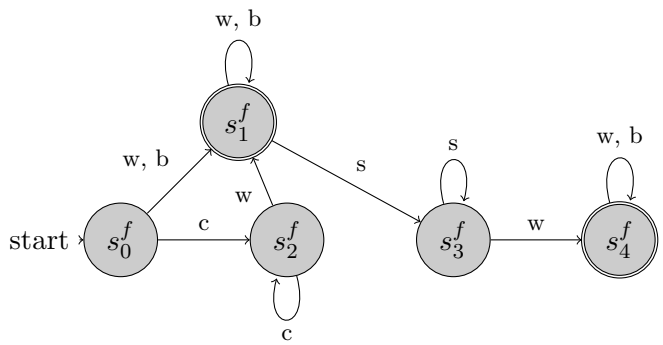
Comparing the different ways of obtaining the backward NFA for general forward NFA would be an interesting follow-up to the present study. In particular the speed-up obtained by optimizing the NFAs would have to be balanced with the CPU time needed to perform the NFA reductions and/or conversion to a minimum-state DFA. Nevertheless, to illustrate the potential gain of optimizing the (backward) NFA, we display, in right part of Figure 4, a possible deterministic finite state automaton for A^b that can we have obtained manually from the reverse regular expression.

The state-based dominance rule applies also for the backward automaton. For the non-deterministic automaton of Figure 4(b), there is no dominance relation between states in the sense of Theorem 5.1. For the deterministic automaton of Figure 4(c), we have $s_1^b \preceq s_4^b$. The set of states that dominate a given forward (backward) state s is denoted $D^f(s)$ ($D^b(s)$), respectively.

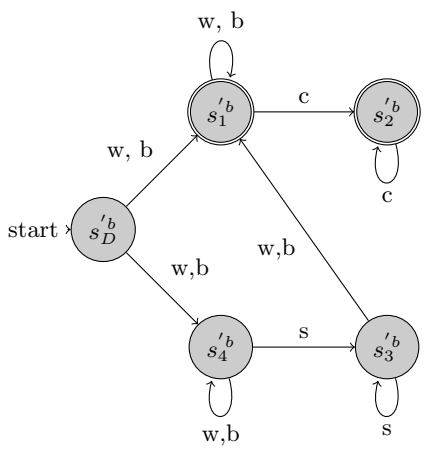
Connection and queue discarding rule

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 case the backward automaton is simply obtained by a reversal of the forward arcs, a forward label (i^f, k^f, s^f) connects to a backward label (i^b, k^b, s^b) if $i^f = i^b$ and $s^f = s^b$ since the backward NFA has the same states as the forward NFA.

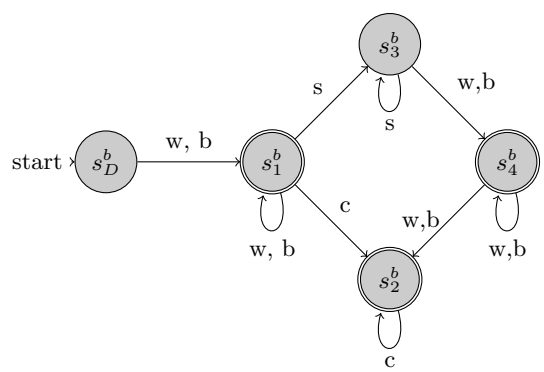
In case the backward automaton has been reduced, a correspondence must be established between the forward and the backward states. Let $CS^{A^b \to A^f}(s)$ (respectively $CS^{A^f \to A^b}(s)$) denote the set of A^f (resp. A^b) states compatible with a given state s of A^b



(a) Deterministic forward automaton



(b) Non-deterministic backward automaton



(c) Deterministic backward automaton

Figure 4: Backward Automata for mode scenario for M_3

(resp. A^b). A forward label (i^f, k^f, s^f) connects to a backward label (i^b, k^b, s^b) if $i^f = i^b$ and $s^f \in CS^{A^b \to A^f}(s^b)$ or, equivalently, $s^b \in CS^{A^f \to A^b}(s^f)$.

For example, with the deterministic backward automaton case of Figure 4(c) and the forward automaton of Figure 4(a), compatible states are: $CS^{A^f \to A^b}(s_1^f) = \{s_1^b, s_4^b\}$ $CS^{A^f \to A^b}(s_2^f) = \{s_2^b\}$, $CS^{A^f \to A^b}(s_3^f) = \{s_3^b\}$ and $CS^{A^f \to A^b}(s_4^f) = \{s_1^b\}$.

We show now that, even if we use the reversed forward automaton for backward search, additional state compatibilities can be automatically established between forward and backward states, so as to have earlier connections without using a manually optimized backward automaton. We first define formally state compatibility:

Definition 5.3. *Forward state s_x^f and backward state s_y^b are compatible if, for any mode m the concatenation of any m-terminated mode string accepted by s_x^f with any reversed m-terminated input string accepted by s_y^b is a valid string.*

We now establish the forward/backward compatibility theorem (sufficient condition).

Theorem 5.2 (Forward/backward state compatibility)**.** *Forward state s_x^f and backward state s_y^b are compatible if $s_x^f \preceq s_y^f$ in the forward automaton or if $s_y^b \preceq s_x^b$ in the backward automaton.*

Proof. If one of the dominance relation holds, the concatenation of any m-terminated

mode string accepted by s_x^f with any reversed m-terminated input string accepted by s_y^b is a valid string, yielding the desired compatibility. □

In our example, there are no state-based dominance in the non-deterministic backward automaton but dominance $s_1^f \preceq s_4^f$ holds in the forward one. Hence, since any input string accepted by s_4^f is accepted by s_1^f, it follows that s_1^b and s_4^f are compatible states, on which a connection can be established.

Note that if dominance relations are represented by a $|S| \times |S|$ boolean matrix, all compatibility relations can be precomputed in $O(|S|^2)$ time.

In case of a connection, the interest of the multi-queue implementation appears. Indeed, when a connection is made between a label (i^f, h, s^f) and a label (i^b, q, s^b) such that the state s^f of FA is compatible with state s^b of BA and

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

holds, all priority queues $FQ_{k'}$ and $BQ_{k'}$ with $k' \geq h + q$ can be discarded since the right hand side is a lower bound of the shortest path with at least $h + q$ transfers

5.5.2 Algorithm description

Algorithm FB-MQLS pseudo-code is given in Appendix B (Algorithm 5). $\mathcal{FQ}$ 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 $k_{\max}$. $BestCurrentSol_k$ stores the best known solution with k transfers and k^* corresponds to the number of transfers of the current best known solution.

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, denoted by (i, s, k). The minimum time label (i, s, k) is then removed from its priority queue (FQ_k or BQ_k) and the forward or backward label extension principle is performed in a similar way as the MQLS Label extension.

The main difference is that (for instance in the forward extension), for each new non-dominated 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 yields 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 as $BestCurrentSol_{k'+k''}$, to possibly update it. Moreover, if the given forward label (j, s', k') is connected with a marked-up backward label with 0 transfer then label (j, s', k') can be discarded as all non-dominated shortest paths from D to i have been found. The forward label extension algorithm is given in Algorithm 6. The backward label extension algorithm is symmetrical and not given in the paper.

After the forward or backward label extension, the queue discarding rule compares the minimum time $BestCurrentSol_{k^*}$ obtained among already established connections, 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, $BestCurrentSol_{k^*}$ 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.

We show in Appendix B the execution of this algorithm for the example of Figure 2. The algorithms marks 8 labels and reaches 22 labels.

5.5.3 Complexity

Compared to MQLS, there is a computational overhead (only at label extension) induced by connection search (step 14 of Algorithm 6). 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 $\tilde{s}.k_{\max}$ where $\tilde{s} = \max_{s \in S^F} |CS^{FA \to BA}(s)| \leq |S^B|$. Hence for the basic dominance rule, the complexity of the forward extension (Algorithm 6)) is $O(|FS_i|(k_{\max} + \log(n|S^F|))$ for the deterministic backward automaton and $O(|FS_i|(k_{\max}\tilde{s} + \log(n|S^F|))$ for the non-deterministic backward automaton. For the state based dominance rule, the complexity of forward extension is $O(|FS_i|(k_{\max}|S^F| + \log(n|S^F|))$ for the deterministic automaton and $O(|FS_i|(k_{\max}(|S^F| + \tilde{s}) + \log(n|S^F|))$ for the non-deterministic backward automaton. If the forward automaton is assumed to be deterministic, the complexity of the backward extension is always equal to $O(|BS_i|(k_{\max} + \log(n|S^B|))$ for the basic dominance rule and to $O(|FS_i|(k_{\max}(|S^B|) + \log(n|S^B|))$ for the state-based dominance rule.

5.6 Goal-oriented techniques with state-dependent estimated travel times

Extension of algorithms TLS and MQLS using the A^* principle (goal oriented search) is straightforward. An interesting possibility is to use a state-dependent heuristic. More precisely, for each label (i, s, k), a modified travel cost denoted by $\tilde{t}^k_{is}$ is given by $t^k_{is} + h^k_{is}$ where t^k_{is} is the travel time from the origin O to i and h^k_{is} is an estimation of the total travel time from i to the destination D, starting in state s with a number of performed transfers equal to k. $\tilde{t}^k_{is}$ represents an estimation of the shortest path from 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^k_{is} correspond to the euclidian distance from i to D over the maximal speed of the modes that can still be taken according to current state s and number of transfers k. Then, h^k_{is} is a lower bound of the minimum travel time from (i, s, k) to any label (D, s', k') in the expanded graph, which insure the optimality of the A^* search. Algorithms TLS and MQLS can be directly extended to A^* principle by using $\tilde{t}^k_{is}$ instead of t^k_{is}.

An example of state-dependent heuristic can be to take the maximum average car speed in state s_2 to compte estimated travel times while in state s_1, the times are computed according to the maximum average bus speed which is slower due to frequent stops.

6 Computational Experiments

The experimental comparisons were carried out on a network covering a part of the urban area of Toulouse (France). Considered modes are $\{w, b, c, s\}$ (walking, bus, private car, subway) and we evaluate three different mode scenarios as previously stated: $M_1 = \{w, b\}$ (walk and bus), $M_2 = \{w, b, s\}$ (walk, bus and subway) and $M_3 = \{w, b, c, s\}$ (all modes).

Viability constraints are modeled by the automaton of Figure 3. Table 1 details the different layers of the transportation network in terms of modes, nodes and arcs. Timetables

for buses and subway are approximate by an average travel time for each corresponding arc in the network.

Table 1: Network data

Modes	**Nodes**	**Arcs**
Bus	3 085	6 646
subway	38	72
Street	59 896	146 280
Transfer	-	6 370
Parking	29	-
Total	63 048	159 368

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. Experiments concern 100 randomly generated origin-destination pairs. Origin and destination are both chosen only in the street subnetwork and around the "center" of the city to avoid border effects. The minimum euclidean distance between them is set to 5 km. The maximum number of transfers is set to $k_{\max} = 10$.

6.1 Results

In Appendix D, Tables 3–8 present the impact of the dominance rules and of the different algorithms on the average CPU time and number of touched nodes needed to solve each of the 100 instances, with the three different NFA configurations. Figure 10 synthesizes the results in terms of average CPU time while Figure 11 concentrates on the number of touched nodes. We first give the characteristics of the obtained solutions and, next, we analyze the performance of the algorithms.

Found solutions

Table 2 presents the average results obtained by all algorithms on the three configurations in terms of average number of solutions, average and maximum number of transfers, minimum and maximum travel times.

Table 2: Solution characteristics

Modes	Nb. Sol.	nb Transf. av	nb Transf. max	Travel t. min	Travel t. max
$M_3 = \{w, b, c, s\}$	3.49	2.7	5.02	48	273
$M_2 = \{w, b, s\}$	4.24	3.59	6.72	62	273
$M_1 = \{w, b\}$	4.23	3.64	6.6	69	273

We obtain from 2 to 6 non dominated solutions per instance with the smallest average value for the most complex automaton. The average number of transfers follows the same trend, so the possibility of using car in the itineraries reduces the number of non-dominated solutions and transfers.

We obtain much more efficient solutions in average than Gräbener *et al.* in [10] for the same objectives, even when the car is used. 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 (parkings) and we have viability constraints. The minimum travel times are in average rather large to ensure a sufficiently large search space end exhibit differences between the algorithms. The larger itineraries correspond to full-walking paths which are always non-dominated as they are the only solutions that induce no transfers (origin and destination being set on the walking street network). The minimum number of transfers is consequently always 0. The average maximal number of transfers may seem large, but a transfer denotes a mode change including walking between stations, so for example a typical bus-subway path yields string $wbwsw$ which involves 4 transfers while the user takes only two modes besides unavoidable walking parts.

Performance of algorithms and dominance rules without viability constraints

We first analyse the results of the different algorithms (TLS, MQLS, their goal oriented variants TLS-A*/MQLS-A* and FB-MQLS) with and without the basic dominance rule on the configuration involving mode scenario $M_1 = \{w, b\}$ and no viability constraints. As there is no automaton, the maximal number of labels (nodes in the expanded graph) is $(n_b + n_w)k_{\max} = 629\,810$. Detailed results are given in Tables 3–4 and on top of Figures 10–11 (Appendix D). Table 3 displays the running time of each combination algorithm/dominance rule in ms while Table 4 gives the number of touched nodes for each combination. In addition, Tables 3 and 4 give in row "Δ b. d." the average speed-up and touched node-decrease obtained when using the basic dominance rule compared to the simple Bellman condition. Row "Δ FB" provides the average speed-up (touched node-decrease) of the bidirectional algorithm compared with other algorithms, using the basic dominance rule.

In this configuration, the problem is precisely the bi-objective problem studied by Pallottino and Scutellà [23]. The TLS algorithm with the basic dominance rule combination can be considered as the reference algorithm, as it resorts to the method proposed by Pallottino and Scutellà [23]. It solves the instances in 132 ms on average with 150 162 touched labels. The basic dominance rule is very efficient as it generates speed-ups from 36% (on TLS) to 45% (on FB-MQLS). The bidirectionnal search is at least 24% faster than the unidirectional algorithms, including the A* variants. The speed-up brought by A* search on unidirectional search is significant but remains modest (12% speed-up for TLS). Note that the speed-ups brought by A* and the state-based dominance rule closely follow the percentage of decrease nodes (15% on average for A* and 37% on average for the basic dominance rule). In the bidirectional search, the need to manage the set of priority queues and to establish connections between several labels having different numbers of transfers induces a computational overhead which explains that the drastic reduction of touched nodes (52% on average) is not fully reflected by the obtained speed-up (41% on average). For the same reasons, the MQLS algorithm has a lower number of touched nodes than the TLS algorithm while it is 11% slower.

In conclusion, the proposed bidirectional FB-MQLS algorithm, used in conjunction with the basic-dominance rule, improves significantly the reference TLS algorithm (45% speed-up yielding an average running time of 89 ms with 64 362 reached labels).

Performance of algorithms and dominance rules with the 3-state NFA

We consider now the mode scenario M_2 (walk, bus and subway) with the 3-state NFA (Tables 5–6 and middle of Figures 10–11). The potential number of labels (i, s, k) now reaches

$(n_s + 2(n_b + n_w))k_{\max} = 1,260\,000$. In addition to TLS, MQLS, and their A* variant, we consider two versions of the FB-MQLS algorithm: FB-MQLS-ND denotes the FB-MQLS algorithm with the non deterministic backward automaton presented in Figure 4(b) while FB-MQLS-D denotes the variant with the deterministic backward automaton presented in Figure 4(c). In Tables 5 (6), we give in row "Δ s. d." the average speed-up (touched node-decrease) obtained when using the state-based dominance rule compared to the basic dominance condition. Row "Δ FB-ND (b. d.)" corresponds to the average speed-up or slowdown (touched node-decrease or increase) of the bidirectional algorithm with the basic dominance rule compared with other algorithms (also using the basic dominance rule) while row "Δ FB-ND (s. d.)" give the same value using the state-based dominance rule.

We observe, independently of the considered dominance rule, that rank FB-MQLS < TLS-A* <MQLS-A*< TLS <MQLS (in terms of increasing running times) is conserved. As expected, a general slowdown is observed for all combinations algorithm/dominance rule. The TLS algorithm with the basic dominance rule now solves the instances in 200 ms in average (34% slower than for the preceding configuration) with a number of touched nodes equal to 218 233. Globally, switching from the basic to the state based dominance rule yields speed-ups for all algorithms from 5% (on FB-MQLS-ND) to 23% (on MQLS). The larger improvement is for the mono-directional algorithms (always more than 20% speed-up) while the bidirectional search is less positively impacted (max 11% speed-up). The speed-up brought by the bidirectional search is also important : it runs from 24% to 41% faster than the unidirectional algorithms. The fastest algorithm if the FB-MQLS-D which in-turn yields an additional speed-up of 25% to FB-MQLS-ND, when used with the state-based dominance rule. Hence, despite the additional forward/backward state compatibilities and dominance rules, optimizing the automaton still yields significant improvements. We also have to underline the leveling effect of the state based dominance rule on the speed-up brought by the bidirectional algorithm. Last, the A* search improves the unidirectional algorithms more than for the previous mode configuration, with an average speed-up of 19%.

In terms of the number of touched nodes, we observe the same phenomenon as for the previous mode configuration: MQLS(-A*) obtains a lower number of touched nodes than TLS(A-*) but the computational overhead is not compensated by the node reduction. For the state-based dominance rule, the reduction of the number of nodes (19% on average for the unidirectionality algorithms) yields the equivalent speed-up (21% on average). As also observed in the previous configuration, FB-MQLS-ND with the basic dominance rule yields an important average touched node decrease compared to the unidirectional algorithms of 52% while it yields "only" a 38% average speed-up.

In conclusion, the proposed bidirectional algorithm (without optimization of the backward automaton) in conjunction with the new state-based-dominance rule improves significantly the reference TLS algorithm with the basic dominance rule (41% speed-up, running time 117 ms, 87 175 reached labels). The speed-up reaches 53% with the optimized backward automaton which reaches an average running time of 94 ms with only 62 936 touched labels.

Performance of algorithms and dominance rules with the 4-state NFA

Tables 7–8 and the bottom part of Figures 10–11 show the results on the 4-state NFA and full mode scenario M_3. The potential number of labels (i, s, k) is equal to $(n_m + n_c + 2(n_w + n_b))k_{\max} = 1\,859\,250$. All the combinations algorithms/dominance rules of

the previous configuration are tested with two variants of the A* algorithms. TLS-A* and MQLS-A* denote the variant where the estimated travel time for a node i to destination D is a constant (as for the previous configurations), whereas TLS-SD-A* and MQLS-SD-A* denote the variant where the estimated travel time for a node i to destination D is state-dependent (see Section 5.6). Our reference algorithm, TLS with basic dominance rule, solves all instances in 215 ms in average with 239 499 touched labels which corresponds to a very moderate slowdown compared to the 3-state automaton. Globally the efficiency of the state-based dominance rule decreases even if the speed-ups are still significant for unidirectional algorithms (always more than 8.5%). However the efficiency of the bidirectional search is reduced. The speed-up compared to TLS and MQLS with the basic dominance rule is significant (17% and 26%, respectively). However, the A* search variants and the bidirectional search are very close. The state-dependent version of the A* search with the state-based dominance rule is even slightly faster than FB-MQLS. The leveling effect of the state-based dominance rule is also important since speed-ups compared to TLS never exceed 10%. We observe a positive but modest effect of the state-based A* search on CPU time compare to state-independent variants.

In conclusion, for the more complex automaton, the leveling effect of the basic and, to a lower extent, state-based dominance rules has to be underlined. The reference TLS algorithm with the basic dominance rule is still 17% slower than the bidirectional algorithm (without optimization of the backward automaton) with the state-based dominance rule which obtains a running time of 177 ms with 133 210 touched nodes. The improvement reaches 20% with the optimized backward automaton (160 ms and 114 370 touched nodes).

Although these times could be certainly reduced with further acceleration mechanisms, our experiments allow to answer positively to the question whether the BI-RegL-CSPP can be solved efficiently on a real urban network.

7 Concluding remarks and further extensions

We have proposed several algorithms to solve the single-source, single-destination bi-objective multi-modal 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. 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. On the largest configuration (an expanded graph of 1 859 250 nodes), we obtain a significant number of non-dominated shortest paths (3.5 on average) with average running times lower than 180 ms. The proposed state-based dominance rules allow to reduce significantly both the CPU times and the number of visited labels for all algorithms. We also observe that the basic and the state-based dominance rule have a positive leveling effect on the algorithm performance. The proposed bidirectional algorithm yields significant speed-ups for most configurations. This speed-up is sensitive to the complexity of the backward automaton. It results that benefits could be obtained by optimizing both forward and backward NFA. A goal-oriented version (A*) of the unidirectional algorithms has been developed, allowing better improvements when a state-based estimated travel time function is used.

A first extension would be to consider time-dependent travel times. A (fast) extension of bidirectional search to time-dependent shortest path (in mono-modal networks) has

been recently proposed by Nannicini *et al.* [19]. The concepts used in their study can be transposed without difficulty in our context under the FIFO assumption. For the forward or mono-directional search, we may simply replace $t + d_{ij}$ by a function $a_{ij}(t)$ which gives the arrival time at j when departing from i at t. 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 can be computed for each encountered connection by a traversal of the backward path given the arrival time at the connection node.

A second extension would be to design an efficient goal oriented scheme for the bidirectional search but a difficulty is to design an efficient queue discarding rule.

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 (ALT, contraction hierarchies) can be implemented. Kirchler *et al* [11] recently proposed an efficient single-objective algorithm following these directions. Furthermore, as suggested by a referee, it is possible that implementation of the ideas presented in this paper with a label-correcting version with an appropriate data structure (perhaps partitioning or double-ended queue) could also yield improvements.

Other multi-objective problems in the multi-modal context are of interest and will be the subject of further research, although the complexity of the problem could increase. The case where public transportation 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 Pallottino in [20] for a single public transportation mode. This approach has been extended to the bi-objective "expected travel time"/"number of transfers" multi-modal viable networks in [16] but no computational experiments where reported. Adapting our algorithms to the hypergraph model is also a promising research direction.

Acknowledgements

The authors wish to thank Arnaud Fradin for his participation to the test campaign and implementation as well as Sandra Ulrich Ngueveu, Roberto Wolfler Calvo, Dominik Kirchler and Emmanuel Néron for fruitful discussions. The authors would also like to thank the referees for their valuable suggestions that contributed greatly to improve the quality of the paper. The research was partially supported by ANRT CIFRE research grant 424/2007 and by GdR Recherche Opérationnelle ARFRO.

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 multi-modal travel systems. *European Journal of Operational Research*, 175(3):1705–1730, 2006.

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

[6] 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.

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

[8] D. Delling, T. Pajor and D. Wagner. Accelerating multi-modal route planning by access nodes. *Algorithms - ESA 2009, Lecture Notes in Computer Science*, 5757:587–598, 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 multi-modal urban routing *Electronic Notes in Discrete Mathematics*, 36:487–494, 2010.

[11] D. Kirchler, L. Liberti, T. Pajor, R. Wolfler Calvo UniALT for regular language constrained shortest paths on a multi-modal transportation network *11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems - ATMOS 2011*, Saarbrücken, Germany, OASICS 20:64–75, 2011.

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

[13] L. Ilie, G. Navarro and S. Yu On NFA reductions in *Theory is forever, Lecture Notes in Computer Science* 3113:112–124,2004.

[14] 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.

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

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

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

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

[19] 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.

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

[21] 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.

[22] T. Pajor. Multi-modal route planning. *Master thesis*, Universität Karlsruhe, Institut für Theoretische Informatik, 2009.

[23] 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.

[24] G. Schnitger. Regular Expressions and NFAs Without $\epsilon-$Transitions. *STACS 2006, Lecture Notes in Computer Science* 2884:432–443, 2006.

[25] 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.

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

[27] 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 A: Comments on the Lozano and Storchi [15] approach

In [15], the set of considered mode is $\{c, s, tr, o\}$ where c i the private mode (car or motorbike), s is the subway mode, tr is a special mode representing a modal transfer and o represents other modes like bus and walking. In their example (similar to our scenario M_3) only c and s are constrained mode in the sense of modal restrictions. Figure 5 represents the corresponding NFA as presented in [15][2].

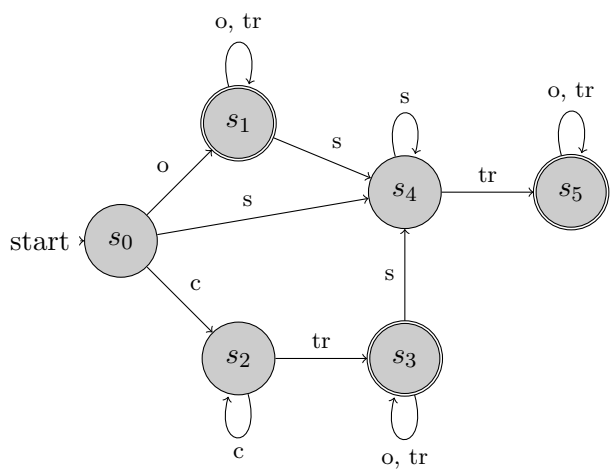


Figure 5: Lozano and Storchi NFA

In their paper Lozano and Storchi proposed a dominance rule (Preference theorem). We recall below the Lozano and Storchi Theorem[3].

Theorem 7.1 ([15]). *State s_u is preferred to state s_l, $s_u \preceq s_l$, if the following sentences are valid:*

(a) the constrained modes used in state s_u path, are a subset of the set of constrained modes used in state s_l path;

(b) if state s_l path is using a constrained mode, then state s_u path is using this mode or has never used it.

To clarify this theorem, we define the following terms. A *"constrained mode"* is a mode which is subject to constraints, i.e. private car (c) and subway (s) in Figure 5. A *"state s path"* is the $O-i$ path represented by a label (i, s, k). *"A mode m has been used" by a "state s path"* means that at least one node j with $m_j = m$ is in the path $O - i$ corresponding to label (i, s, k). A *"state s path is using mode m"* means that for label (i, s, k), $m_i = m$.

The topological Pallottino and Scutellà [23] algorithm was extended by Lozano and Storchi to path viability modeled by a NFA using the preference theorem as a dominance rule for label discarding. The algorithm (MSVPP) iterates on the number of transfers from

[2]In [15] impossible transitions were also represented but they can be guessed with the observation that any non represented transition is infeasible, e.g. subway, bus or waking cannot be taken from state s_2

[3]We replace notation R, used in the original theorem, by notation $\preceq$

0 to $k_{\max}$. At each iteration k, it searches for the shortest path with exactly k transfers. For that, the data structure Q for storing labels, is made of two priority queues Q^{now} and Q^{next}, where Q^{now} contains labels with k transfers and Q^{next} contains labels with $k+1$ transfers. Initially, Q^{now} contains only label (O, s_0) while Q^{next} is empty. At a typical iteration, a label (i, s) is taken from Q^{now} and its value t_{is} is compared with $lastlabel_{is}$, the shortest travel time found so far to reach node i in state s (i.e. with at most k transfers). If $t_{is} < lastlabel_{is}$, the label extension principle is applied and $lastlabel_{is}$ is updated. Each successor (j, s') such that $t_{is} + d_{ij} < t_{js'}$ is candidate for being updated. A dominance rule based on the preference theorem is applied. If label (j, s') has a longest length than *all* labels (j, s'') for all states s'' that are preferred to s', label (j, s') is discarded. Otherwise, $t_{js'}$ is updated with $t_{is} + d_{ij}$. If arc (i, j) is not a transfer arc, the new label (j, s') is inserted in Q^{now}. If arc (i, j) is a transfer arc, the label is inserted in Q^{next}. When Q_{now} is empty, the number of transfers is incremented. Q_{now} is reinitialized with Q_{next}. If Q_{now} is empty or if a maximum number of transfers is reached, the algorithms stops. Note that MSVPP is a label correcting algorithm. Note also that, in this algorithm, a single label t_{is} is used per node-state pair (i, s), while Pallottino and Scutellà [23] use two labels per nodes, which would give the two labels t_{is}^{now} and t_{is}^{next}.

We now point out the following issues linked to the Lozano and Storchi preference theorem, the way the dominance rule is implemented and the way the topological Pallottino and Scutellà [23] algorithm was extended. We refer to the NFA displayed in Figure 5.

- The set of "constrained modes" used by each label is not fully determined by the label state.

  Even if, for state s_1, the set of used constrained modes is necessarily empty and if, for states s_2 and s_3, the set of used constrained mode is necessarily singleton $\{c\}$, the set of constrained modes used by a label in state s_4 or s_5 can be one of the sets $\{s\}$ or $\{s, c\}$. Consequently, to apply dominance rule of Theorem 7.1 for two labels (i, s) and (i, s'), one should either build the partial path by a recursively accessing the predecessor labels p_{is} and $p_{is'}$ to access the visited modes or, better, one may store, for each label, the set of visited constrained modes. However, both solutions raise computational issues.

- The application of the dominance rule to discard a label (i, s) in the Lozano and Storchi algorithm may consider a label as non-dominated while it could be discarded.

  Consider the simple network of Figure 6 and the bi-objective shortest path problem from node x_0 to node x_6. Consider also the three paths issued from x_0 and ending at x_5. We have private car-bus path $p^{(1)}$: $x_0 \to x_1 \to x_3 \to x_5$ with 1 transfer and length 4 (shortest path). We have direct bus path $p^{(2)} : x_0 \to x_5$ with no transfer and length 6 (shortest path with 0 transfer). Last, we have subway-bus path $p^{(3)} : x_0 \to x_2 \to x_4 \to x_5$ with 1 transfer and length 5. Path $p^{(3)}$, entering node x_5 in state s_5, is dominated by path $p^{(1)}$, entering node x_5 in state s_4, as it has the same number of transfers, a larger length and less extension possibilities. Hence it should be discarded. In the Lozano and Storchi algorithm, path $p^{(2)}$, entering node x_5 in state s_1, is first found as the shortest path with 0 transfers. Suppose $p^{(1)}$ is first one found as the shortest path to x_5 with 1 transfer. When the dominated path $p^{(3)}$ is scanned, it is unfortunately not discarded as, according to the way the dominance rule is applied in the Lozano-Storchi algorithm, a label can only be discarded if it has a length larger than or equal to the one of *all* its preferred labels, not just one as

in classical dominance checking. According to the preference theorem, both labels $p^{(1)}$ and $p^{(2)}$ are preferred to $p^{(3)}$ (as they both have more extension possibilities). However $p^{(3)}$ has shortest length than $p^{(2)}$ and so the algorithm does not discard it.

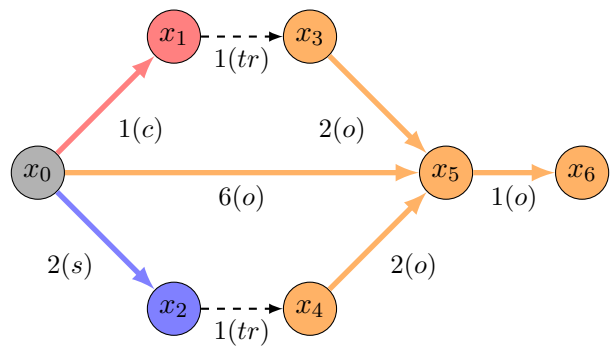


Figure 6: A dominated path is not discarded

- Theorem 7.1 is not valid anymore if the NFA is slightly modified.

  Consider the simple multimodal network displayed in Figure 7 and the NFA obtained by removing transition $s_1 \xrightarrow{s} s_4$ from the NFA of Figure 5. This may model the (unusual) preference that if the private car or the metro are not taken from the origin, the metro cannot be taken in any viable path. Hence, path $x_0 \to x_2 \to x_3 \to x_5 \to x_6 \to x_4 \to x_7$ with 2 transfers and a length of 12 is not feasible.

  The shortest path is $x_0 \to x_1 \to x_2 \to x_3 \to x_5 \to x_6 \to x_4 \to x_7$ of length 12 and 3 transfers corresponding to mode sequence private car, bus, subway, bus. The 0-transfer path $x_0 \to x_2 \to x_3 \to x_4 \to x_7$ taking only the bus has a length of 13. Let us apply the preference theorem to subpaths p $(x_0 \to x_2)$ and p' $(x_0 \to x_1 \to x_2)$. p is in state s_1 while p' is in state s_3. p has used no constrained modes while p' has used constrained mode p. Consequently $s_1 \preceq s_3$ for p' and p. As p is reached by the algorithm before p', when p' is scanned by the algorithm it is discarded since it does not improve the length of any previously found path reaching x_2 in a preferred state (p is the only path with a state preferred to that of p'). Since p' is discarded, arc (x_3, x_5) will never be viable as node x_3 (in the modified automaton) will only be reached in state s_1 by extending p and that taking the subway is no more possible in state s_1. As the shortest path will never be found, the dominance rule of Theorem 7.1 is not valid for this (slightly) modified NFA. Actually the dominance rule is valid for the example given in [15], as the subway can be reached by both states s_1 and s_3 but our simple counterexample shows that it cannot be directly generalized to different state transitions.

- A path with k transfers may not be extended because it has a longer length than a path with $k+1$ transfers.

  To illustrate this situation, let us consider the example given in Figure 8. When computing the shortest path from x_0 to x_4 with 0 transfer, the algorithm obtains labels (x_1, s_1) with $t_{x_1 s_1} = 1$ and (x_2, s_4) with $t_{x_2 s_4} = 2$ and stores them in Q_{now}. Then, (x_1, s_1) is dequeued and (x_3, s_4) with 1 transfer is obtained and stored in Q_{next}, updating $t_{x_3 s_4} = 2$. And, then (x_2, s_4) is dequeued from Q_{now}. Label (x_3, s_4) with 0 tranfer is obtained but is travel time is equal to 4 which is greater than the

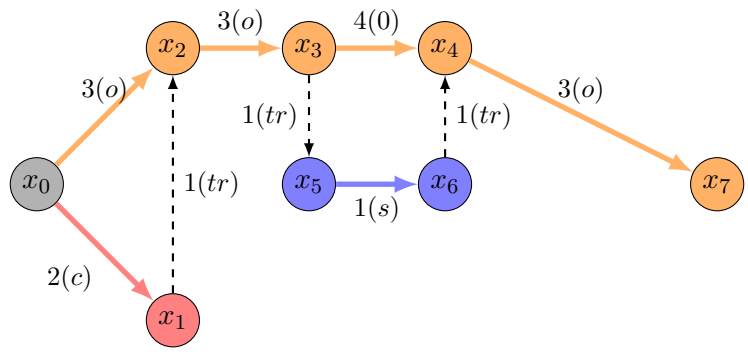


Figure 7: The shortest path is discarded if the state automaton is changed

previous value just obtained for (x_3, s_4). It follows that label (x_3, s_4) with 0 transfer is discarded. And, the algorithm doesn't find any solution with 0 transfer. The problem comes from the fact that, unlike in the original algorithm of Pallottino and Scutellà [23], a single label t_{is} is used to update any label independently of the fact that is is stored in Q_{now} and in Q_{next}.

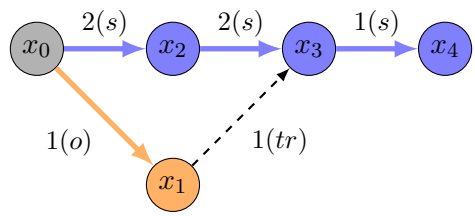


Figure 8: The shortest path from x_0 to x_4 with 0 tranfer is not obtained

In this paper, we address all these issues by proposing a new state-based dominance rule and a different implementation of the TLS algorithm.

Appendix B: Algorithm pseudo-code

Topological label setting algorithm (TLS)

Algorithm 1 Topological label-setting algorithm (TLS)

Require: Graph: $G(V, E)$, NFA: A, Origin: O, Destination: D, Max. transfers: $k_{\max}$

```
{Initialization}
Q^now ← {(O, s_0, 0)}, t^0_{O,s_0} ← 0, p^0_{O,s_O} ← (O, s_0, 0), Q^next ← ∅
t^k_{is} ← ∞, ∀i ∈ V \ {O}, ∀s ∈ S, ∀k = 0, ..., k_max
k ← 0
{Search shortest path from 0 to k_max transfers}
while Q^now ≠ ∅ and k ≤ k_max do
    repeat
        (i, s, k) ← argmin{t^k_{js'} | (j, s', k) ∈ Q^now} and Q^now ← Q^now \ {(i, s, k)}
        if (i ≠ D or s ∉ F) and t^k_{is} <_BestLastSol then
            LabelExtensionTLS((i, s, k), Q^now, Q^next)
        end if
    until Q^now = ∅ or (i = D and s ∈ F)
    if i = D and s ∈ F and t^k_{is} <_BestLastSol then
        BestLastSol ← t^k_{is}
        store t^k_{Ds} and p^k_{Ds} (shortest path with k transfers).
    end if
    k ← k + 1, Q^now ← Q^next and Q^next ← ∅
end while
```

Algorithm 2 LabelExtensionTLS((i, s, k), Q^{now}, Q^{next})

```
Mark^k_{is} ← true
for j ∈ FS(i) do
    for s' ∈ δ(m_i, m_j, s) do
        if m_i = m_j then
            k' ← k
        else
            k' ← k + 1
        end if
        if not Mark^{k'}_{js'} then {Scan unmarked successor (j, s', k')}
            if t^k_{is} + d_ij < t^{k'}_{js'} then {Bellman condition is not satisfied}
                t^{k'}_{js'} ← t^k_{is} + d_ij, p^{k'}_{js'} ← (i, s, k)
                if k' = k then
                    BestValue_{js'} ← t^{k'}_{js'}
                end if
                if ∀s'' ∈ D_{s'}, t^k_{is} + d_ij < BestValue_{js''} then {Dominance rule does not apply}
                    if k' = k then
                        Q^now ← Q^now ∪ {(j, s', k')}
                    else
                        Q^next ← Q^next ∪ {(j, s', k')}
                    end if
                end if
            end if
        end if
    end for
end for
```

Multi-queue label setting algorithm (MQLS)

Algorithm 3 Multi-queue label setting algorithm (MQLS)

Require: Graph: $G(V, E)$, NFA: A, Origin: O, Destination: D, Max. transfers: $k_{\max}$

$\mathcal{Q} = \{Q_0 \leftarrow \{(O, s_0, 0)\}\}$, $t^0_{0,s_0} \leftarrow 0$, $p^0_{O,s_0} \leftarrow (O, s_0, 0)$
$t^0_{i,s} \leftarrow \infty$, $\forall i \in V \setminus \{O\}, \forall s \in S$
$K \leftarrow k_{\max}$
repeat
  $(i, s, k) \leftarrow argmin\{t^{k'}_{i',s'} | (i', s', k') \in \mathcal{Q}\}$ and $Q_k \leftarrow Q_k \setminus \{(i, s, k)\}$
  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 $K \leftarrow k - 1$
  else
    LabelExtensionMQLS((i, s, k), $\mathcal{Q}$)
  end if
until $K < 0$ or $\mathcal{Q} = \emptyset$

Algorithm 4 LabelExtensionMQLS((i, s, k), $\mathcal{Q}$)

$Mark^k_{is} \leftarrow true$
for $j \in FS(i)$ **do**
  for $s' \in \delta(m_i, m_j, s)$ **do**
    if $m_i = mj$ **then**
      $k' \leftarrow k$
    else
      $k' \leftarrow k + 1$
    end if
    if not $Mark^{k'}_{js'}$ **then**
      if $t^k_{is} + d_{ij} < t^{k'}_{js'}$ **then** {*Bellman condition does not hold*}
        if $\forall s'' \in \mathcal{D}_{s'}$, $\forall k'' \leq k'$, $t^k_{is} + d_{ij} < t^{k''}_{js''}$ **then** {*Dominance rule does not apply*}
          $t^{k'}_{js'} \leftarrow t^k_{is} + d_{ij}$, $p^{k'}_{js'} \leftarrow (i, s, k)$
          $Q_{k'} \leftarrow Q_{k'} \cup \{(j, s', k')\}$
        end if
      end if
    end if
  end for
end for

Bidirectionnal multi-queue label setting algorithm (FB-MQLS)

Algorithm 5 Bidirectional multi-queue label setting algorithm (FB-MQLS)

Require: Graph $G(V, E)$, NFA: FA, BA, Origin: O, Destination: D, Max. transfers: $k_{\max}$

{Initial forward and backward labels}

$\mathcal{FQ} \leftarrow \{FQ_0 \leftarrow \{(O, s_0, 0)\}\}$, $ft^0_{O,s_0} \leftarrow 0$, $fp^0_{O,s_0} \leftarrow (0, s_0, 0)$

$ft^0_{i,s} \leftarrow \infty$, $\forall i \in V$, $\forall s \in S^F$, $(i, s) \neq (O, s_0)$

$\mathcal{BQ} \leftarrow \{BQ_0 \leftarrow \{(D, s_D, 0)\}\}$, $bt^0_{D,s_D} \leftarrow 0$, $fp^0_{D,s_D} \leftarrow (D, s_D, 0)$

$bt^0_{i,s} \leftarrow \infty$, $\forall i \in V$, $\forall s \in S^B$, $(i, s) \neq (D, s_D)$

{Initial minimal connection}

$K \leftarrow k_{\max}$, $BestCurrentSol_k \leftarrow \infty, \forall k, 0 \leq k \leq k_{\max}$, $k^* \leftarrow -1$

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

repeat

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

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

  {Direction setting}

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

    $(i, s, k) \leftarrow (i^f, s^f, k^f)$, $FQ_k \leftarrow FQ_k \setminus \{(i^f, s^f, k^f)\}$

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

      $BestCurrentSol_k \leftarrow ft^k_{is}$, $k^* \leftarrow k$

    else

      ForwardLabelExtension$((i, s, k), \mathcal{FQ}, \mathcal{BQ}, k^*, BestCurrentSol)$

    end if

  else

    $(i, s, k) \leftarrow (i^b, s^b, k^b)$, $BQ_k \leftarrow BQ_k \setminus \{(i^b, s^b, k^b)\}$

    if $i = O$ and $s \in F^B$ **then**

      $BestCurrentSol_k \leftarrow bt^k_{is}$, $k^* \leftarrow k$

    else

      BackwardLabelExtension$((i, s, k), \mathcal{FQ}, \mathcal{BQ}, k^*, BestCurrentSol)$

    end if

  end if

  {queue discarding test}

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

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

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

    store $BestCurrentSol_{k^*}$ as the shortest path with k^* transfers.

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

    $K \leftarrow k^* - 1$, $k^* \leftarrow argmin_{k \in \{0,\ldots,K\}} BestCurrentSol_k$

  end if

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

Algorithm 6 ForwardLabelExtension($(i, s, k), \mathcal{FQ}, \mathcal{BQ}, k^*$,$BestCurrentSol$)

$ForwardMark_{is}^{k} \leftarrow true$
for $j \in FS(i)$ **do**
  for $s' \in \delta(m_i, m_j, s)$ **do**
    if $m_i = mj$ **then**
      $k' \leftarrow k$
    else
      $k' \leftarrow k + 1$
    end if
    if not $ForwardMark_{js'}^{k'}$ **then**
      if $ft_{is}^{k} + d_{ij} < ft_{js'}^{k'}$ **then** {Bellman condition does not hold}
        if $\forall s'' \in \mathcal{D}_{s'}^{F}$, $\forall k'' \leq k'$, $ft_{is}^{k} + d_{ij} < ft_{js''}^{k''}$ **then** {Dominance rule does not apply}
          $ft_{js'}^{k'} \leftarrow ft_{is}^{k} + d_{ij}$, $fp_{js'}^{k'} \leftarrow (i, s, k)$
          {Checking connections with backward labels}
          $insertheap \leftarrow true$
          for $(j, s^b, k'') \in \mathcal{BQ} \cup \mathcal{M}$, $k' + k'' \leq K$, $s^b \in CS^{FA \to BA}(s')$ **do**
            if $BestCurrentSol_{k'+k''} > ft_{j,s'}^{k'} + bt_{j,s^b}^{k''}$ **then**
              $BestCurrentSol_{k'+k''} \leftarrow ft_{j,s'}^{k'} + bt_{j,s^b}^{k''}$
              if $k^* = -1$ or $BestCurrentSol_{k'+k''} < BestCurrentSol_{k^*}$ **then**
                $k^* \leftarrow k' + k''$
              end if
              if $k'' = 0$ and $BackwardMark_{j,s^b}^{k''} = true$ **then**
                $insertheap \leftarrow false$ {Connection with marked backward with 0 transfer}
              end if
            end if
          end for
          if $insertheap$ **then**
            Insert$((j, s', k'), FQ_{k'})$
          end if
        end if
      end if
    end if
  end for
end for

Appendix C: Algorithm illustration

7.1 Topological Label Setting (TLS)

Initialization: $k = 0$, $Q^{now} = \{(x_1, s_0, 0)\}$, $t_{1s_0}^{0} = 0$, $Q^{next} = \emptyset$, $k_{\max} = 5$

Iteration 1: $k = 0$ // Search solution with 0 transfer

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_1, s_0, 0)$; $t_{1s_0}^{0} = 0$	$(x_4, s_0, 0)$	4	-	Q^{now}
	$(x_2, s_0, 1)$	1	-	Q^{next}
	$(x_6, s_1, 1)$	1	-	Q^{next}
$(x_4, s_0, 0)$; $t_{4s_0}^{0} = 4$	$(x_5, s_0, 0)$	8	-	Q^{now}
	$(x_3, s_0, 1)$	5	-	Q^{next}
	$(x_7, s_1, 1)$	5	-	Q^{next}
$(x_5, s_0, 0)$; $t_{5s_0}^{0} = 8$	Destination is reached			

$Q^{now} = \{(x_2, s_0, 1), (x_6, s_1, 1), (x_3, s_0, 1), (x_7, s_1, 1)\}$, $t_{2s_0}^{1} = 1, t_{6s_1}^{1} = 1, t_{3s_0}^{1} = 5, t_{7s_1}^{1} = 5$, $Q^{next} = \emptyset$

Iteration 2: $k = 1$ // Search solution with 1 transfer

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_2, s_0, 1)$; $t^1_{2s_0} = 1$	$(x_3, s_0, 1)$	6		already in Q^{now}
	$(x_4, s_0, 2)$	2	-	Q^{next}
$(x_6, s_1, 1)$; $t^1_{6s_1} = 1$	$(x_4, s_2, 2)$	6	state	-
	$(x_7, s_1, 1)$	4		already in Q^{now}
$(x_7, s_1, 1)$; $t^1_{7s_1} = 4$	$(x_5, s_2, 2)$	5		Q^{next}
$(x_3, s_0, 1)$; $t^1_{3s_0} = 5$	$(x_5, s_0, 2)$	6		Q^{next}
Q^{now} is empty : no solution with 1 transfer.				

$Q^{now} = \{(x_4, s_0, 2), (x_5, s_2, 2), (x_5, s_0, 2)\}$, $t^2_{4s_0} = 2, t^2_{5s_2} = 5, t^2_{5s_0} = 6$, $Q^{next} = \emptyset$

Iteration 3: $k = 2$ // Search solution with 2 transfers

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_4, s_0, 2)$; $t^2_{4s_0} = 2$	$(x_5, s_0, 2)$	6		already in Q^{now}
	$(x_3, s_0, 3)$	3		Q^{next}
	$(x_7, s_1, 3)$	3		Q^{next}
$(x_5, s_2, 2)$; $t^2_{5s_2} = 5$	Destination is reached			

$Q^{now} = \{(x_3, s_0, 3), (x_7, s_1, 3)\}$, $t^3_{3s_0} = 3, t^3_{7s_1} = 3$, $Q^{next} = \emptyset$

Iteration 4: $k = 3$ // Search solution with 3 transfers

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_3, s_0, 3)$; $t^3_{3s_0} = 3$	$(x_5, s_0, 4)$	3		Q^{next}
$(x_7, s_1, 3)$; $t^3_{7s_1} = 3$	$(x_5, s_2, 4)$	4	state	-
Q^{now} is empty : no solution with 3 transfers.				

$Q^{now} = \{(x_5, s_0, 4)\}$, $t^4_{5s_0} = 4$, $Q^{next} = \emptyset$

Iteration 5: $k = 4$ // Search solution with 4 transfers

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_5, s_0, 4)$; $t^4_{5s_0} = 4$	Destination is reached			

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.

Multi-queue label setting algorithm (MQLS)

Initialization: $k = 0$, $Q_0 = \{(x_1, s_0, 0)\}$, $t^0_{1s_0} = 0$, $k_{\max} = 5$

fixed label ; travel time	reached label	travel time	Dom. rule	Insert
$(x_1, s_0, 0)$; $t^0_{1s_0} = 0$	$(x_4, s_0, 0)$	4	-	Q_0
	$(x_2, s_0, 1)$	1	-	Q_1
	$(x_6, s_1, 1)$	1	-	Q_1
$(x_2, s_0, 1)$; $t^1_{2s_0} = 1$	$(x_3, s_0, 1)$	6	-	Q_1
	$(x_4, s_0, 2)$	2	-	Q_2
$(x_6, s_1, 1)$; $t^1_{6s_1} = 1$	$(x_7, s_1, 1)$	4	-	Q_1
	$(x_4, s_2, 2)$	2	state	-
$(x_4, s_0, 2)$; $t^2_{2s_0} = 2$	$(x_5, s_0, 2)$	6	-	Q_2
	$(x_3, s_0, 3)$	3	-	Q_3
	$(x_7, s_1, 3)$	3	-	Q_3
$(x_3, s_0, 3)$; $t^3_{3s_0} = 3$	$(x_5, s_0, 4)$	4	-	Q_4
$(x_7, s_1, 3)$; $t^3_{7s_1} = 3$	$(x_5, s_2, 4)$	4	state	-
$(x_5, s_0, 4)$; $t^4_{5s_0} = 4$	Destination is reached. Suppress Q_4			
$(x_7, s_1, 1)$; $t^1_{7s_1} = 4$	$(x_5, s_2, 2)$	5	-	Q_2
$(x_4, s_0, 0)$; $t^0_{4s_0} = 4$	$(x_5, s_0, 0)$	8		Q_0
	$(x_3, s_0, 1)$	5	basic	-
$(x_3, s_0, 1)$; $t^1_{3s_0} = 5$	$(x_5, s_0, 2)$	6	basic	-
$(x_5, s_2, 2)$; $t^2_{5s_2} = 5$	Destination is reached. Suppress Q_2 and Q_3			
$(x_5, s_0, 0)$; $t^0_{5s_0} = 8$	Destination is reached. Suppress Q_0 and Q_1			

The algorithm stops as the solution with 0 transfer has been obtained.

Bidirectional Multi-queue label setting algorithm (FB-MQLS)

For the proposed bidirectional algorithm, we consider the NFA given in Figure 9. Compatible states correspond to

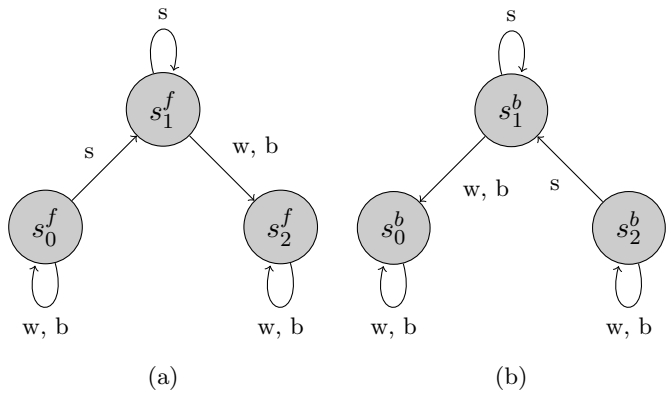


Figure 9: NFA for forward and backward steps

$CS^{FA\to BA}(s_0^f) = \{s_D^b, s_2^b\}$, $CS^{FA\to BA}(s_1^f) = \{s_1^b\}$ and $CS^{FA\to BA}(s_2^f) = \{s_2^b\}$. For the forward NFA, there is a dominance relation : $s_0^f \preceq s_2^f$ and for the backward one, $s_2^b \preceq s_D^b$.

Initialization: $k^* = -1$, $FQ_0 = \{(x_1, s_0^f, 0)\}$, $ft^0_{1s_0^f} = 0$, $BQ_0 = \{(x_5, s_2^b, 0)\}$, $bt^0_{5s_2^b} = 0$, $k_{\max} = 5$. To simplify the following tables, variable $BestCurrentSol_k$ is denoted by $Best_k$.

Direction (F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
F; $(x_1, s_0^f, 0)$; $ft^0_{1s_0^f} = 0$	$(x_4, s_0^f, 0)$	4	-	-	FQ_0
	$(x_2, s_0^f, 1)$	1	-	-	FQ_1
	$(x_6, s_1^f, 1)$	1	-	-	FQ_1
Queue Dicarding test: Not applicable					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
B; $(x_5, s_2^b, 0)$; $bt^0_{5s_2^b} = 0$	$(x_4, s_2^b, 0)$	4	-	$(x_4, s_0^f, 0)$, $Best_0{=}8$, $k^* = 0$	BQ_0
	$(x_3, s_2^b, 1)$	1	-	-	BQ_1
	$(x_7, s_1^b, 1)$	1	-	-	BQ_1
Queue Dicarding test: $Best_0{=}8 \leq 1{+}0$: No					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
F; $(x_2, s_0^f, 1)$; $ft^1_{2s_0^f} = 1$	$(x_3, s_0^f, 1)$	6	-	$(x_3, s_2^b, 1)$, $Best_2{=}7$, $k^* = 2$	FQ_1
	$(x_4, s_0^f, 2)$	2	-	$(x_4, s_2^b, 0)$, $Best_2{=}6$, $k^* = 2$	FQ_2
Queue Dicarding test: $Best_2{=}6 \leq 1{+}1$: No					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
F; $(x_6, s_1^f, 1)$; $ft^1_{6s_1^f} = 1$	$(x_7, s_1^f, 1)$	4	-	$(x_7, s_1^b, 1)$, $Best_2{=}5$, $k^* = 2$	FQ_1
	$(x_4, s_2^f, 2)$	2	State	-	
Queue Dicarding test: $Best_2{=}5 \leq 2{+}1$: No					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
B; $(x_3, s_2^b, 1)$; $bt^1_{3s_2^b} = 1$	$(x_2, s_2^b, 1)$	6	-	$(x_2, s_0^f, 1)$, no improvement	FQ_1
	$(x_4, s_2^b, 2)$	2	-	$(x_4, s_0^f, 0)$, no improvement	
				$(x_4, s_0^f, 0)$, $Best_4{=}4$, $k^* = 4$	FQ_1
Queue Dicarding test: $Best_4{=}4 \leq 2{+}1$: No					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
B; $(x_7, s_1^b, 1)$; $bt^1_{7s_1^b} = 1$	$(x_6, s_1^b, 1)$	4	-	$(x_6, s_1^f, 1)$, no improvement	BQ_1
	$(x_4, s_0^b, 2)$	2	State	-	-
Queue Dicarding test: $BestCurrentSol_4{=}4 \leq 2{+}2$: Yes. Shortest path with 4 transfers Suppress FQ_4 and BQ_4. $k^* = 2$					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
F; $(x_4, s_0^f, 2)$; $ft^2_{4s_0^f} = 2$	$(x_5, s_0^f, 2)$	6	-	$(x_5, s_2^b, 0)$, no improvement	- (mark-up label)
	$(x_3, s_0^f, 3)$	3	-	$(x_3, s_2^b, 1)$, too many transfer	FQ_3
	$(x_7, s_1^f, 3)$	3	-	$(x_7, s_1^b, 1)$, too many transfer	FQ_3
Queue Dicarding test: $Best_2 = 5 \leq 4+2$: Yes. Shortest path with 2 transfers Suppress FQ_2, FQ_3, BQ_2 and BQ_3. $k^* = 0$					

(F/B); fixed label ; travel time	reached label	travel time	Dom. rule	Connection	Insert
F; $(x_4, s_0^f, 0)$; $ft^0_{4s_0^f} = 4$	$(x_5, s_0^f, 0)$	8	-	$(x_5, s_2^b, 0)$, no improvement	- (mark-up label)
	$(x_3, s_0^f, 1)$	5	-	$(x_3, s_2^b, 1)$, too many transfer	FQ_1
	$(x_7, s_1^f, 1)$	5	-	$(x_7, s_1^b, 1)$, too many transfer	FQ_1
Queue Dicarding test: $Best_0 = 8 \leq 6+4$: Yes. Shortest path with 0 transfers Suppress FQ_0, FQ_1, BQ_0 and BQ_1. $k^* = -1$					

The algorithm stops as the solution with 0 transfer has been obtained.

Appendix D: detailed computational results

Table 3: Impact of dominance rule and algorithms on CPU time (ms) - no NFA

	FB-MQLS	TLS (A*)	MQLS (A*)	TLS	MQLS
Bell. cond.	160	183	197	206	236
Basic dom.	87	116	124	132	149
Δb. d.	-45%	-37%	-37%	-36%	-37%
Δ FB	0%	-24%	-29%	-34%	-41%

Table 4: Impact of dominance rule and algorithms on touched nodes- no NFA

	FB-MQLS	MQLS (A*)	TLS (A*)	MQLS	TLS
Bell. cond	114660	188605	207423	220818	229972
Basic dom	64362	120828	132664	141762	150162
Δb. d.	-44%	-36%	-36%	-36%	-35%
Δ FB	0%	-47%	-51%	-55%	-57%

Table 5: Impact of dominance rules and algorithms on CPU time (ms) - 3 states

	FB-MQLS (Det)	FB-MQLS (Non Det)	TLS (A*)	MQLS (A*)	TLS	MQLS
Bell. cond	180	215	272	297	312	379
Basic dom	106	123	169	190	200	243
State dom	94	117	134	150	157	186
Δs. d.	-11%	-5%	-20%	-21%	-21%	-23%
Δ FB-ND (b. d.)	16%	0%	-27%	-35%	-39%	-50%
Δ FB-ND (s. d.)	25%	0%	-13%	-22%	-25%	-37%

Table 6: Impact of dominance rules and algorithms on touched nodes - 3 states

	FB-MQLS (Det)	FB-MQLS (Non Det)	MQLS (A*)	TLS (A*)	MQLS	TLS
Bell. cond	120475	155711	259103	293863	318538	331905
Basic dom	70862	91940	167473	187768	206055	218233
State dom	62936	87175	137457	152955	166399	176261
Δs. d.	-11%	-5%	-18%	-19%	-19%	-19%
Δ FB-ND (b. d.)	30%	0%	-45%	-51%	-55%	-58%
Δ FB-ND (s. d.)	39%	0%	-37%	-43%	-48%	-51%

Table 7: Impact of dominance rules and algorithms on CPU time (ms) - 4 states

	FB-MQLS (Det)	FB-MQLS (Non Det)	TLS (sd-A*)	MQLS (sd-A*)	TLS (A*)	MQLS (A*)	TLS	MQLS
Bell. cond	220	244	309	318	334	339	352	410
Basic dom	162	178	186	195	201	208	215	241
State dom	160	177	170	180	184	191	194	218
Δs. d.	-1%	-1%	-9%	-8%	-9%	-8%	-10%	-10%
Δ FB-ND (b. d.)	10%	0%	-4%	-9%	-12%	-14%	-17%	-26%
Δ FB-ND (s. d.)	10%	0%	4%	-2%	-4%	-7%	-9%	-19%

Table 8: Impact of dominance rules and algorithms on touched nodes - 4 states

	FB-MQLS (Det)	FB-MQLS (Non Det)	MQLS (sd-A*)	TLS (sd-A*)	MQLS (A*)	TLS (A*)	MQLS	TLS
Bell. cond	154829	180813	294543	305298	340893	346405	362986	374727
Basic dom	117908	136458	194345	199614	213322	217390	228648	239499
State dom	114370	133210	180266	184742	196596	200105	209325	218997
Δs. d.	-3%	-2%	-7%	-7%	-8%	-8%	-8%	-9%
Δ FB-ND (b. d.)	16%	0%	-30%	-32%	-36%	-37%	-40%	-43%
Δ FB-ND (s. d.)	16%	0%	-26%	-28%	-32%	-33%	-36%	-39%

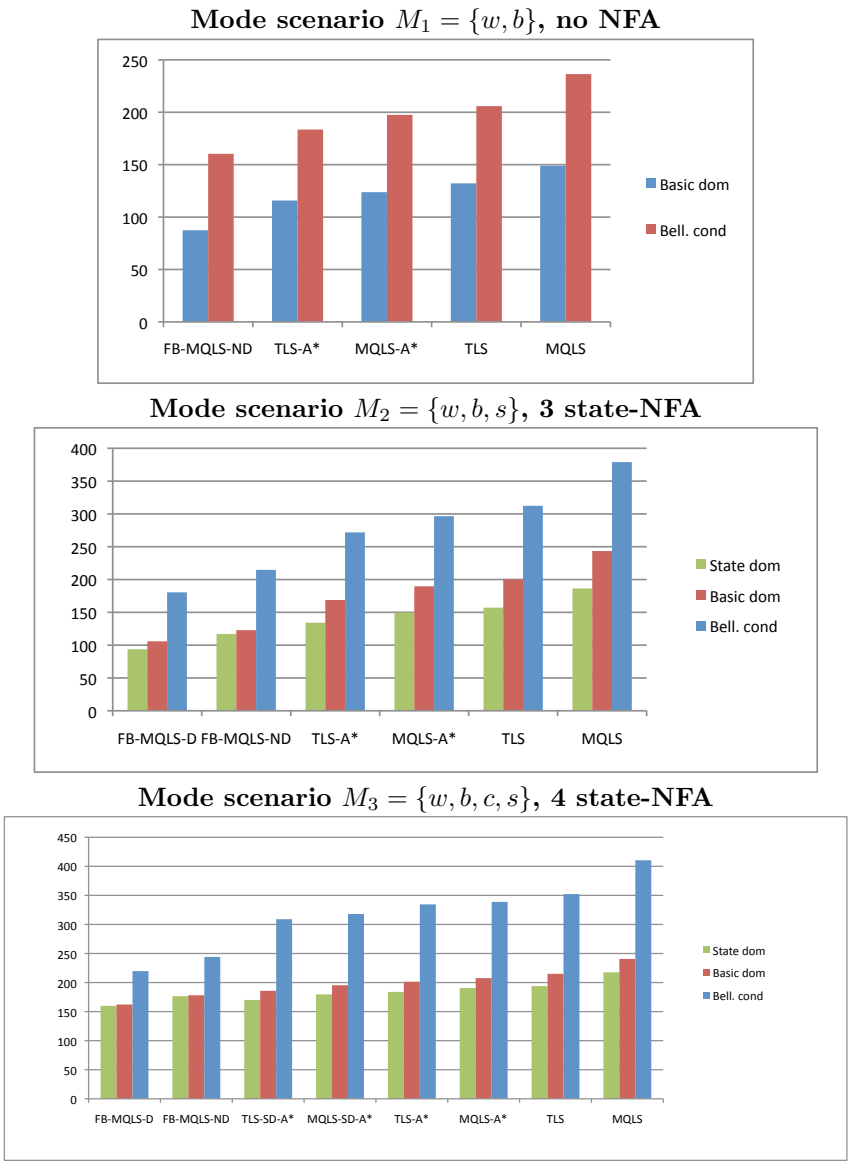


Figure 10: Impact of dominance rules and algorithms on CPU time (ms) for different NFAs

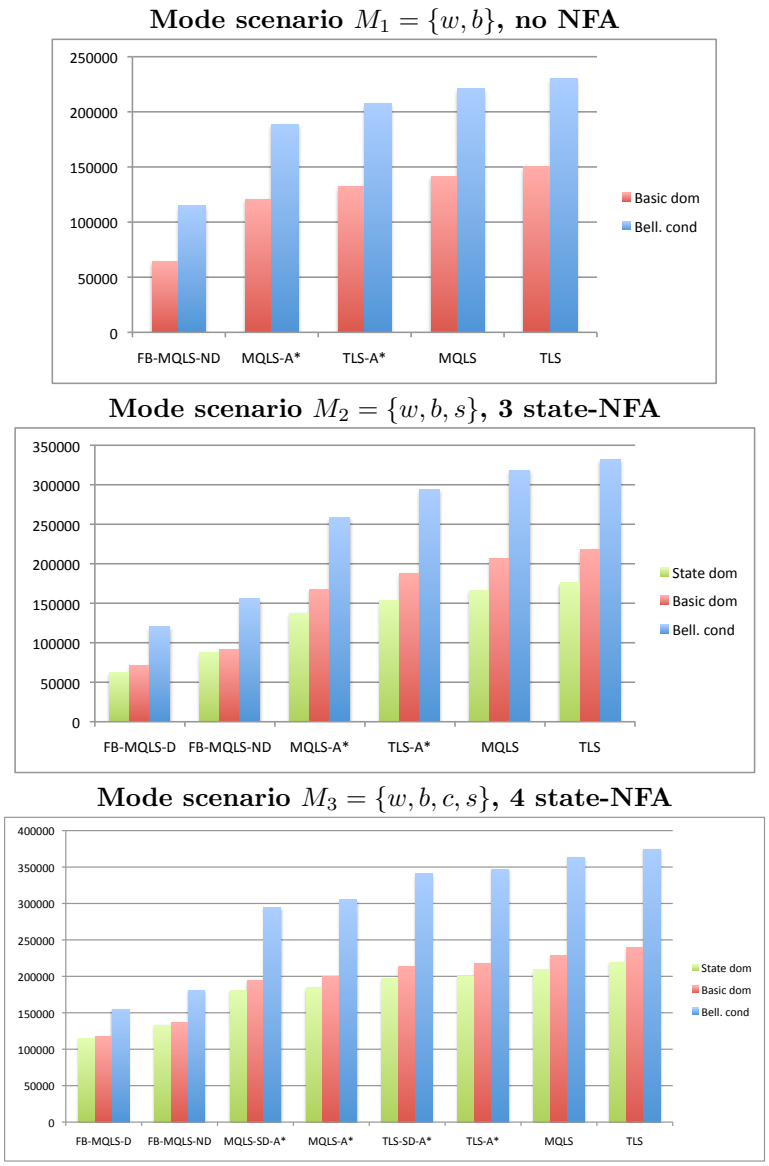


Figure 11: Impact of dominance rules and algorithms on touched nodes for different NFAs