[EURO Journal on Computational Optimization 9 (2021) 100004](https://doi.org/10.1016/j.ejco.2021.100004)

Contents lists available at [ScienceDirect](http://www.ScienceDirect.com/)

EURO Journal on Computational Optimization

journal homepage: [www.elsevier.com/locate/ejco](http://www.elsevier.com/locate/ejco)

The Steiner bi-objective shortest path problem

Hamza Ben Ticha[a](#_bookmark0), Nabil Absi [a](#_bookmark0),[∗](#_bookmark2), Dominique Feillet[a](#_bookmark0), Alain Quilliot[b](#_bookmark1)

a *Ecole des Mines de Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CMP, Departement SFL, F-13541, Gardanne, France*

b *LIMOS, Institut Supérieur d’Informatique de Modélisation et leurs Applications, ISIMA, Campus des Cèzeaux, Aubière Cedex, France*

a r t i c l e i n f o a b s t r a c t

*Keywords:*

A∗ algorithm

Multiple destinations

Vehicle routing with road-network information

graph *𝐺* = (*𝑉 , 𝐴*)*,* with a subset *𝑇 ⊂ 𝑉* of terminals. Arcs are labeled with travel time and cost. The goal is to find In this paper, we introduce the Steiner Bi-objective Shortest Path Problem. This problem is defined on a directed a complete set of eﬃcient paths between every pair of nodes in *𝑇* . The motivation behind this problem stems from

with a multi-objective A∗ search strategy guiding the search towards the terminals. Computational results based data preprocessing for vehicle routing problems. We propose a solution method based on a labeling approach

on instances generated from real road networks show the eﬃciency of the proposed algorithm compared to state-of-art approaches.

# Introduction

The Vehicle Routing Problem (VRP) can be described as the prob- lem of designing a set of routes that start and end at a depot and that visit a number of geographically dispersed locations, called customers. In the standard version of the problem, the road network of the geo-

graph *𝐺* = (*𝑇 , 𝐷*) is introduced, where *𝑇* is composed of the depot and graphic area at hand is not explicitly considered. Instead, a directed the customers, and *𝐷* represents all the possible connections between these nodes: *𝐷* = {(*𝑖, 𝑗*) ∶ *𝑖* ∈ *𝑇 , 𝑗* ∈ *𝑇* ⧵ {*𝑖*}}. A weight *𝑐𝑖𝑗* is then associ- ated with every arc (*𝑖, 𝑗*) ∈ *𝐷* to indicate travel costs (distances) between

nodes.

Weights *𝑐𝑖𝑗* are assumed to be precomputed using the road-network

structure, which can very easily be done with shortest path algorithms.

for each arc in *𝐷* (*e.g.*, algorithm A∗, Hart et al. [Hart et al. (1968)](#_bookmark42)), by This can be performed by applying a goal directed search independently solving one-to-all shortest paths starting from each node in *𝑇* (*e.g.*, Di-

jkstra’s algorithm, [Dijkstra (1959)](#_bookmark35)), or by computing all-to-all shortest paths (*e.g.*, Floyd’s algorithm, [Floyd (1962)](#_bookmark36)). All these algorithms admit a polynomial-time complexity and generally allow a very fast compu- tation of the data in view of the limited number of customers in VRP applications (rarely more than a few hundreds). A large amount of liter- ature exists to accelerate these algorithms for large-scale networks (see, *e.g.*, [Bast et al. (2016)](#_bookmark60)), but none of them are really necessary in this context.

In many real-world routing problems however, and in most variants of the VRP, this model is not accurate enough to determine optimal so- lutions ([Ben Ticha et al., 2018](#_bookmark30)). Indeed, road segments usually have at

least two attributes, time and distance, and nodes in *𝑇* may be connected

together using many different paths with different trade-offs in distance and time. The VRP with Time Windows (VRPTW) gives a good illustra- tion of this diﬃculty. In the VRPTW, vehicle routes are constrained by time windows that define the earliest and latest possible starting times of the service for customers. The objective is to minimize the total traveled

distance. The min-distance path between any pair of customers *𝑖* and *𝑗*

from *𝑖* to *𝑗* on the cost function. However, depending on the arrival time should usually be preferred because it minimizes the impact of traveling to *𝑖,* a fastest path could sometimes be required to reach customer *𝑗* on

time. As it depends on the complete sequence of customers in the route, deciding which path should be preferred between two successive cus- tomers cannot be determined in advance. The preprocessing approach described above is thus not appropriate.

To handle this situation, a more effective model has been proposed by [Garaix et al. (2010)](#_bookmark38). It consists in representing road network infor-

mation with a multigraph. The node set is still *𝑇* but the arc set is re-

placed by an arc multiset *𝐷* constructed as follows. Given two nodes in *𝑇 ,* a complete set of eﬃcient paths is computed between these two nodes. Then, an arc is added to *𝐷* for every path in the complete set.

Arc weights (travel distance, travel time) are defined to their value in the corresponding path. An example of a multigraph is depicted on [Figure 1](#_bookmark3). In this small example, pairs of nodes are connected by at most two arcs: in some cases the min-cost and min-time paths coin- cide, in other cases they are distinct and two parallel arcs are added. More generally, the size of the Pareto front, *i.e.* the number of Pareto optimal objective vectors, can be much larger and many parallel arcs could be inserted. All these eﬃcient solutions are essential because they might all offer the best compromise to minimize costs and satisfy time constraints.

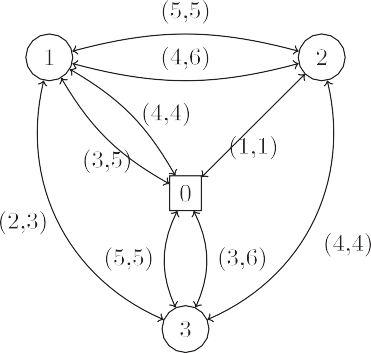
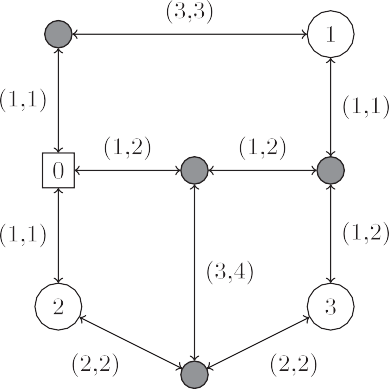
∗ Corresponding author.

*E-mail addresses:* [hamza.ben-ticha@emse.fr](mailto:hamza.ben-ticha@emse.fr) (H. Ben Ticha), [absi@emse.fr](mailto:absi@emse.fr) (N. Absi), [feillet@emse.fr](mailto:feillet@emse.fr) (D. Feillet), [alain.quilliot@isima.fr](mailto:alain.quilliot@isima.fr) (A. Quilliot).

<https://doi.org/10.1016/j.ejco.2021.100004>

Received 30 August 2020; Received in revised form 3 March 2021; Accepted 3 March 2021

2192-4406/© 2021 The Author(s). Published by Elsevier Ltd on behalf of Association of European Operational Research Societies (EURO). This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>)

**Fig. 1.** Road-network graph and multigraph for a simple example with one depot, three cus- tomers and two attributes (arcs labeled with (dis- tance,time)).



This new model opens important research perspectives that re- cently started receiving a strong interest (*e.g.*, [Lai et al. (2016)](#_bookmark46), [Letchford et al. (2014)](#_bookmark47), [Huang et al. (2017)](#_bookmark43), [Ben Ticha et al. (2017)](#_bookmark61)). Among others, it opens the question of the tractability of computing the multigraph. Clearly, the arc multiset and their weights cannot be ob- tained by solving simple shortest path problems. The aim of the Steiner Multi-objective Shortest Path Problem is to deliver this information. It

of nodes in *𝑇* . In what follows, we adopt the standard terminology of consists in computing complete sets of eﬃcient paths between every pair Steiner tree problems and call the nodes in *𝑇* terminals.

In this paper, we introduce the Steiner Multi-objective Shortest Path Problem and propose a solution approach for its solution. We limit our study to the case of two attributes (Steiner Bi-objective Shortest Path Problem, Steiner BSPP). The proposed approach is based on a dynamic

programming algorithm with an A∗ guiding strategy. In addition to the

introduction of this new problem, the contributions of the paper are:

* An original implementation of the A∗ strategy that simultaneously

drives the search towards all the terminals.

* An experimental study giving new insights on how eﬃciently multi- graph data can be computed for vehicle routing problems.

The rest of this paper is organized as follows. In [Section 2](#_bookmark4), we present an overview of the related literature. [Section 3](#_bookmark7) formally introduces the Steiner BSPP and useful notation. It also reports some basic properties that will be helpful for the remainder of the paper. In [Section 4](#_bookmark8), we describe our solution method. In [Section 5](#_bookmark13), we propose some enhance- ments when the problem is addressed in the context of the VRPTW. Finally, numerical results are reported in [Section 6](#_bookmark16).

# Literature review

The Steiner BSPP is related to two types of problems: bi-objective shortest path problems (BSPPs) and “Steiner-like” problems:

* BSPPs are multi-objective extensions of standard shortest path prob- lems (SPPs). Every arc of the graph receives two weights (that cor- respond to two criteria) and the goal is to generate eﬃcient paths. Three variants can naturally be considered: the one-to-one BSPP, the one-to-all BSPP and the all-to-all BSPP, where eﬃcient paths are searched between one origin and one destination, one origin and all destinations, all origins and all destinations, respectively. All three variants are NP-hard ([Serafini, 1987](#_bookmark58)).

lem. Given a weighted graph *𝐺* = (*𝑉 , 𝐴*) and a set of terminals *𝑇 ⊂ 𝑉 , •* Steiner problems originate from the well-known Steiner Tree Prob-

all the nodes in *𝑇* . It generalizes the minimum spanning tree prob- the Steiner tree problem consists in finding a min-cost tree covering

lem, where all nodes have to be covered. Following this terminology, the Steiner Traveling Salesman Problem was introduced and inves- tigated by several authors ([Cornuéjols et al., 1985; Letchford et al.,](#_bookmark34)

tices in *𝑇 ⊂ 𝑉* is sought. It generalizes the Traveling Salesman Prob- [2013](#_bookmark34))). In this problem, a minimum cost cycle visiting all the ver- lem (case *𝑇* = *𝑉* ). Following the same trend, we called our problem

Steiner BSPP because we are only interested in shortest paths be-

paths. It generalizes the all-to-all shortest path problem (*𝑇* = *𝑉* ) and tween a subset of nodes (the terminals) instead of all-to-all shortest the one-to-one shortest path problem (|*𝑇* | = 2)

If the relationship with Steiner problems is important to understand the name given to our problem, the Steiner BSPP is essentially a variant of BSPPs. Actually, solving the all-to-all BSPP also solves the Steiner BSPP. Equivalently, solving a one-to-all BSPP starting from each node

in *𝑇* or solving a one-to-one BSPP for each pair of nodes in *𝑇* also solves

the Steiner BSPP. Clearly our point here is to propose a more eﬃcient

approach.

BSPPs (and more generally Multi-objective SPPs) have been widely studied in the literature, as acknowledged by several literature reviews [(Clímaco and Pascoal, 2012; Ehrgott and Gandibleux, 2000; Skriver, 2000)). A first important matter concerns the size of solution sets, which](#_bookmark32) directly translates to the number of arcs that would have to be added to the multigraph in our VRP context. [Hansen (1980)](#_bookmark39) shows that, in the worst case, this number can be exponential in the size of graph

[*𝐺*.](#_bookmark48) [Müller-Hannemann and Weihe (2006)](#_bookmark51) [and Mandow and Pérez de la](#_bookmark48)

[Cruz (2009) however show that much better behaviors can be observed](#_bookmark48)

in practice. Evaluating the typical size of the solution sets in our context will be a meaningful output of the paper.

A second important information that can be derived from the liter- ature concerns exact solution methods. The prominent approaches to solve BSPPs are labeling algorithms. These algorithms all follow the same principle ([Hansen, 1980; Martins, 1984b](#_bookmark39)). A label represents a par- tial path in the graph. The most elementary operation in the algorithm is to select a label and extend the associated partial path with an additional arc. It results in a new label that can be dominated or kept for further extensions. Algorithms differ in the order in which these elementary la- bel extensions are performed and in the data structures that are used to eﬃciently manage label selection and dominance. Labeling algorithms can solve one-to-one and one-to-all BSPPs. Algorithms applied to solve all-to-all SPPs have a very different nature and, as far as we know, have never been extended to the bi-objective or multi-objective case. Also, the presence of nonpositive arc weights can complicate a lot the problems, [especially when elementary paths are required (see, *e.g.*, Martins and Santos (1999),](#_bookmark52) [Irnich](#_bookmark44) [and Desaulniers (2005) or](#_bookmark52) [Feillet](#_bookmark37) [et al. (2004)).](#_bookmark52) Seeing that we are only interested in positive weights in our context, we focus on this case hereafter.

Regarding the one-to-all BSPP, two main label selection heuris- tics are employed in the literature. In a node-selection strategy *e.g.*, ([Brumbaugh-Smith and Shier, 1989](#_bookmark31)), all labels representing paths end- ing at a same node are selected and extended to all successors of this

node. In a label-selection strategy, a single label is selected, based on its value. Again this label is extended to all the successors of the ending node of the path that it represents (*e.g.*, [Tung and Chew (1992)](#_bookmark59)). Several computational studies evaluated various variants of these two strategies [and gave a slight advantage to node-selection techniques (Guerriero and Musmanno, 2001; Paixão and Santos, 2013).](#_bookmark40)

with a destination-driven selection rule, following the principle of A∗ In the one-to-one BSPP, the label-selection strategy is employed but

[algorithm (Mandow and Pérez de la Cruz, 2010; Stewart and White III, 1991). Labels are selected according to an optimistic evaluation of the](#_bookmark49) cost that paths reaching the destination from this label could have. In order to eﬃciently select labels and add new labels to the set of labels in wait, a heap is generally used. In [Mandow and Pérez de la Cruz (2010)](#_bookmark49), all label evaluations (which are actually composed of a value for each criterion) are compared and a label with a non-dominated evaluation is selected. In [Tung and Chew (1992)](#_bookmark59), the two components of the eval- uation are added and the label with the smallest sum is selected. The evaluation measure is given by the actual cost of the label completed, for both components, by the mono-criterion shortest path cost to reach the destination.

Another category of solution methods is based on ranking. They determine paths progressively, in a non-decreasing order of one of the objectives ([Climaco and Martins, 1982; Raith and Ehrgott, 2009](#_bookmark33)). [Martins (1984a)](#_bookmark50) proposed a ranking method based on path deletion. In this method, the shortest path according to a selected criterion is de- termined at each iteration. This path is then eliminated from the net- work for the next iteration. The algorithm stops when the expected number of path is found or when no more eﬃcient paths are found. [Huarng et al. (1996)](#_bookmark45) however showed that this ranking method was not competitive compared to labeling approaches.

Another approach was proposed by Mote et al. [Mote et al. (1991)](#_bookmark53). It is organized in two phases. In the first phase, extreme solutions in

We introduce the subset *𝑇 ⊂ 𝑉* of terminals. The Steiner BSPP aims at

finding a complete set of eﬃcient paths between every pair of terminals.

A few definitions are given below, to clarify what a complete set of eﬃcient paths is. We consider in these definitions that all paths have the same starting and ending points.

**Definition 1.** *Dominance*

A vector (*𝑎*1 *, 𝑎*2 ) dominates a vector (*𝑏*1 *, 𝑏*2 ) if and only if *𝑎*1 ≤ *𝑏*1 and

*𝑎*2 ≤ *𝑏*2 with at least one inequality being strict.

**Definition 2.** *Efficient path*

A path *𝑃*1 is eﬃcient if there does not exist any path *𝑃*2 whose cost vector (*𝑑*(*𝑃*2 )*, 𝑡*(*𝑃*2 )) dominates (*𝑑*(*𝑃*1 )*, 𝑡*(*𝑃*1 )).

**Definition 3.** *Set of non-dominated cost vectors (also called Pareto front)*

The set of all non-dominated cost vectors is the set of vectors

(*𝑑*(*𝑃* )*, 𝑡*(*𝑃* )) obtained from eﬃcient paths.

**Definition 4.** *Complete set of efficient paths*

vector (*𝑑, 𝑡*) admits at least one path *𝑃* ∈ P such that (*𝑑, 𝑡*) = (*𝑑*(*𝑃* )*, 𝑡*(*𝑃* )). A set P of eﬃcient paths is complete if every non-dominated cost

A complete set is minimal if none of its subsets is complete.

For the remainder of the paper, we denote |*𝑉* | = *𝑛,* |*𝐴*| = *𝑚* and |*𝑇* | =

*𝑛𝑇* .

*3.2. Additional notation and basic properties*

that will be useful in next sections. We first recall that a vector *𝑎* = We now introduce additional notation and some simple properties (*𝑎*1 *, 𝑎*2 ) is lexicographically smaller than a vector *𝑏* = (*𝑏*1 *, 𝑏*2 )*,* denoted by

*𝑎 < 𝑏,* if either *𝑎 < 𝑏* or both *𝑎* = *𝑏* and *𝑎 < 𝑏* . Similarly, we say

1 1 1 1 2 2

*𝑙𝑒𝑥*

that a path *𝑃*1 is lexicographically smaller than a path *𝑃*2*,* denoted by

*𝑃*1 *< 𝑃*2 *,* if and only if (*𝑑*(*𝑃*1 )*, 𝑡*(*𝑃*1 )) *<* (*𝑑*(*𝑃*2 )*, 𝑡*(*𝑃*2 )). The lexicographic

the convex hull of the BSPP solution space are computed by solving

*𝑙𝑒𝑥*

*𝑙𝑒𝑥*

the LP relaxation of the problem. In the second phase, an enumerative method is used to determine the set of eﬃcient paths. The main idea is to restrict enumeration thanks to the information extracted from the first phase. [Raith and Ehrgott (2009)](#_bookmark56) studied more deeply this two-phase ap- proach. They investigated different combinations of methods for the two

order defines a total order between paths.

Let P(*𝑢, 𝑣*) denote the set of all paths linking two nodes *𝑢* and *𝑣* in

*𝑇* and let P*𝑜𝑝𝑡* (*𝑢, 𝑣*) = (*𝑃*1 *, 𝑃*2 *, ..., 𝑃𝑟* ) be a complete set of eﬃcient paths between *𝑢* and *𝑣*. Without loss of generality, we assume that P*𝑜𝑝𝑡* (*𝑢, 𝑣*) is minimal and that paths in P*𝑜𝑝𝑡* (*𝑢, 𝑣*) are sorted according to the lexico-

graphic order, *i.e.*, *𝑃*1 *< 𝑃*2 *< ... < 𝑃𝑟* . As a consequence:

phases: a network simplex method and single objective label setting and

*𝑙𝑒𝑥*

*𝑙𝑒𝑥*

*𝑙𝑒𝑥*

label correcting algorithms were tested in Phase 1 and, ranking and bi- objective labeling approaches were explored in Phase 2. They compared the two-phase method with purely labeling approaches and a ranking method. Computational experiments carried out on different instance sets showed the competitiveness of the two-phase method with the dif- ferent configurations. In their conclusions, [Raith and Ehrgott (2009)](#_bookmark56) no- ticed that the eﬃciency of solution methods depends a lot on network

*𝑑*(*𝑃*1 ) *< 𝑑*(*𝑃*2 ) *< ... < 𝑑*(*𝑃𝑟*−1 ) *< 𝑑*(*𝑃𝑟* )

*𝑡*(*𝑃*1 ) *> 𝑡*(*𝑃*2 ) *> ... > 𝑡*(*𝑃𝑟*−1 ) *> 𝑡*(*𝑃𝑟* )

Using this notation, we can recall the following simple properties:

**Property 1.** *𝑃*1 *is the shortest path in distance from 𝑢 to 𝑣 in 𝐺: 𝑑*(*𝑃*1) =

structure.

*𝑚𝑖𝑛*

*𝑃*

*𝑑*(*𝑃* )*.*

Besides these exact methods, many heuristic algorithms have also been applied to BSPPs, such as evolutionary algorithms (*e.g.*,

∈P(*𝑢,𝑣*)

**Property 2.** *𝑃𝑟 is the shortest path in time from 𝑢 to 𝑣 in 𝐺: 𝑡*(*𝑃𝑟* ) =

[Pangilinan and Janssens (2007)](#_bookmark55)) or ant colony optimization algorithms

*𝑚𝑖𝑛*

*𝑃*

*𝑡*(*𝑃* )*.*

(*e.g.*, [Ghoseiri and Nadjari (2010)](#_bookmark41)). These algorithms are however out of the scope of this research as we focus on exact solution.

∈P(*𝑢,𝑣*)

**Proof.** For each non-eﬃcient path *𝑃* ∈ P(*𝑢, 𝑣*) ⧵ P

*𝑜𝑝𝑡*

(*𝑢, 𝑣*)*,* at least one

# Problem definition, notation and basic properties

* 1. *Problem definition*

We consider a directed graph *𝐺* = (*𝑉 , 𝐴*) modeling a road network. Arcs (*𝑖, 𝑗*) ∈ *𝐴* represent road segments and are tagged with two positive weights *𝑑𝑖𝑗* and *𝑡𝑖𝑗* . We define a path *𝑃* in *𝐺* as an ordered list of nodes

*𝑃* = (*𝑢*0 *, 𝑢*1 *, ..., 𝑢𝑝* ) such that (*𝑢𝑘 , 𝑢𝑘*+1 ) ∈ *𝐴* for *𝑘* ∈ {0*, ..., 𝑝* − 1}. The cost

vector of path *𝑃* is the sum of its arc weights:

path *𝑃𝑘* ∈ P*𝑜𝑝𝑡* (*𝑢, 𝑣*) exists such that *𝑑*(*𝑃𝑘* ) ≤ *𝑑*(*𝑃* ) and *𝑡*(*𝑃𝑘* ) ≤ *𝑡*(*𝑃* ). Con-

sequently, for each path *𝑃* ∈ P(*𝑢, 𝑣*)*,* we have *𝑑*(*𝑃*1) ≤ *𝑑*(*𝑃* ) and *𝑡*(*𝑃𝑟* ) ≤

*𝑡*(*𝑃* ). □

In the remainder of this paper, we denote by (*𝑑𝑚𝑖𝑛*(*𝑢, 𝑣*)*, 𝑡𝑚𝑎𝑥*(*𝑢, 𝑣*)) the cost vector associated with the shortest path in distance in P*𝑜𝑝𝑡* (*𝑢, 𝑣*) and we denote by (*𝑑𝑚𝑎𝑥*(*𝑢, 𝑣*)*, 𝑡𝑚𝑖𝑛*(*𝑢, 𝑣*)) the cost vector associated with the shortest path in time in P*𝑜𝑝𝑡* (*𝑢, 𝑣*). Note that any eﬃcient path *𝑃* ∈ P(*𝑢, 𝑣*) is such that *𝑑𝑚𝑖𝑛*(*𝑢, 𝑣*) ≤ *𝑑*(*𝑃* ) ≤ *𝑑𝑚𝑎𝑥*(*𝑢, 𝑣*) and *𝑡𝑚𝑖𝑛*(*𝑢, 𝑣*) ≤ *𝑡*(*𝑃* ) ≤ *𝑡𝑚𝑎𝑥*(*𝑢, 𝑣*).

# Solution algorithm

(*𝑑*(*𝑃* )*, 𝑡*(*𝑃* )) =

(

∑

*𝑘*∈{0*,...,𝑝*−1}

*𝑑𝑢𝑘 𝑢𝑘*+1

∑

*,*

*𝑘*∈{0*,...,𝑝*−1}

)

*𝑡𝑢𝑘 𝑢𝑘*+1

The core mechanism of our solution algorithm is similar to the label setting algorithm proposed, first, by [Martins (1984b)](#_bookmark54) which is, in turn, based on Dijkstra’s algorithm ([Dijkstra, 1959](#_bookmark35)). It essentially follows a

one-to-all solution framework, with the difference that, in our adapta- tion, we are only interested in the eﬃcient paths arriving to nodes in

*𝑇* and we guide the search towards this direction. To solve the Steiner

BSPP, this algorithm is repeated several times, once for each terminal

defined as a starting point

In what follows, the starting node is named *𝑣*0. A label represents a path *𝑃* from *𝑣*0 to a certain node *𝑢* ∈ *𝑉* and is defined with the following

information:

*𝐿* = (*𝑙𝑎𝑠𝑡*(*𝐿*)*, 𝑑*(*𝐿*)*, 𝑡*(*𝐿*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*))

where *𝑙𝑎𝑠𝑡*(*𝐿*) = *𝑢, 𝑑*(*𝐿*) = *𝑑*(*𝑃* )*, 𝑡*(*𝐿*) = *𝑡*(*𝑃* ) and *𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*) is the la- bel from which *𝐿* was extended (∅ for the initial label). Following previous definitions, we say that a label *𝐿*1 dominates another label

*𝐿*2 if (*𝑑*(*𝐿*1 )*, 𝑡*(*𝐿*1 )) dominates (*𝑑*(*𝐿*2 )*, 𝑡*(*𝐿*2 )). We say that *𝐿*1 is lexico- graphically smaller than *𝐿*2 *,* denoted by *𝐿*1 *< 𝐿*2 *,* if (*𝑑*(*𝐿*1 )*, 𝑡*(*𝐿*1 )) *<*

sociated with shortest paths in distance and, *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑣*) and *𝑡𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*)

indicating distances and times associated with shortest paths in time.

* Using Dijkstra’s algorithms in backwards from all nodes *𝑠* ∈ *𝑇* (that

distance and in time from all nodes *𝑣* ∈ *𝑉* to destination nodes *𝑠* ∈ *𝑇* . is, with the arcs implicitly reversed), we compute shortest paths in Four series of tables are obtained: *𝑑𝑚𝑖𝑛*(*𝑣, 𝑠*) and *𝑡𝑚𝑎𝑥*(*𝑣, 𝑠*) indicating

distances and times associated with shortest paths in distance and,

*𝑑𝑚𝑎𝑥*(*𝑣, 𝑠*) and *𝑡𝑚𝑖𝑛*(*𝑣, 𝑠*) indicating distances and times associated with

shortest paths in time.

isons are based on the lexicographical order *<* . Thanks to that, we have In both forward and backward Dijkstra’s algorithms, label compar-

*𝑙𝑒𝑥*

the guarantee that all the computed paths are eﬃcient.

Algorithm MDA∗ mainly relies on the two following structures:

*𝑙𝑒𝑥*

*𝑙𝑒𝑥*

* + Set *allLabels* contains all the labels that have to be extended. It is

(*𝑑*(*𝐿*2 )*, 𝑡*(*𝐿*2 )); we say that the two labels are equal if (*𝑑*(*𝐿*1 )*, 𝑡*(*𝐿*1 )) =

(*𝑑*(*𝐿*2)*, 𝑡*(*𝐿*2))*,* even if the father nodes can differ.

In [Section 4.1](#_bookmark9), we detail the solution method. In [Section 4.2](#_bookmark12), we

prove that the proposed algorithm correctly provides the expected set of eﬃcient paths. We then give, in [Section 4.3](#_bookmark14), more details on the data structures and report on the complexity.

* 1. *The multi-destination-A*∗ *algorithm*

[Algorithms 1](#_bookmark10) and [2](#_bookmark11). We call this algorithm multi-destination-A∗ algo- The solution method and its preprocessing are described in rithm (MDA∗). It provides a minimal complete set of eﬃcient paths be- tween *𝑣*0 and *𝑠*.

**Algorithm 1** Preprocessing

1: compute *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*), *𝑡𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*), *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑣*) and *𝑡𝑚𝑎𝑥*(*𝑣*0 *, 𝑣*) for all *𝑣* ∈ *𝑉*

2: **for all** *𝑠* ∈ *𝑇* **do**

3: compute *𝑑𝑚𝑖𝑛*(*𝑣, 𝑠*), *𝑡𝑚𝑖𝑛*(*𝑣, 𝑠*), *𝑑𝑚𝑎𝑥*(*𝑣, 𝑠*) and *𝑡𝑚𝑎𝑥*(*𝑣, 𝑠*) for all *𝑣* ∈ *𝑉*

4: **end for**

**Algorithm 2** Multi-destination-A∗ algorithm for the Steiner BSPP

1: *𝐿* = (*𝑣*0*,* 0*,* 0*,* ∅)

2: *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠.𝑎𝑑𝑑*(*𝐿*)

3: *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*0]*.𝑎𝑑𝑑*(*𝐿*)

4: **while** *𝐾*(*𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠.𝑀𝑖𝑛*()) ≤ *𝑚𝑎𝑥𝐾 𝑚𝑎𝑥* **do**

*𝑠*∈*𝑇*

*𝑠*

5: *𝐿* = *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠.𝑒𝑥𝑡𝑟𝑎𝑐𝑡𝑀𝑖𝑛*()

6: *𝑢* = *𝑙𝑎𝑠𝑡*(*𝐿*)

7: **for all** (*𝑢, 𝑣*) ∈ *𝐴* **do**

8: *𝐿*′ = (*𝑣, 𝑑*(*𝐿*) + *𝑑𝑢𝑣 , 𝑡*(*𝐿*) + *𝑡𝑢𝑣 , 𝐿*)

9: **if** *𝐿*′ is not dominated nor equal to a label in *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] **then**

10: *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠.𝑎𝑑𝑑*(*𝐿*′)

11: *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*]*.𝑎𝑑𝑑*(*𝐿*′)

12: **if** a label *𝐿*′′ ∈ *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] is dominated by *𝐿*′ **then**

13: *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠.𝑟𝑒𝑚𝑜𝑣𝑒*(*𝐿*′′)

14: *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*]*.𝑟𝑒𝑚𝑜𝑣𝑒*(*𝐿*′′)

15: **end if**

16: **end if**

17: **end for**

18: **end while**

future labels to the starting position *𝑣*0 (Lines 1–2). When a label used for label selection. It is initialized with a single label anchoring

with a new non-dominated cost vector is created, it is added to this set (Line 10). When a label is selected for extension or dominated, it is removed (Lines 5 and 13)

nodes *𝑣* ∈ *𝑉* . It is used for dominance. It is initially empty for all *•* Vector *Labels[v]* contains all the eﬃcient paths that are known for nodes but *𝑣*0 (Line 3). Labels with new non-dominated cost vectors

are added (Line 11), newly dominated labels are removed (Line 14).

At each iteration, a label *𝐿* is selected in *allLabels* (Line 5) and ex- tended to all the successors of *𝑙𝑎𝑠𝑡*(*𝐿*) (Line 7). If new promising la-

bels are found (Line 9), the different label sets are updated as explained above.

to lead the most quickly to one of the nodes in *𝑇* . This is where the Label selection function *extractMin()* aims at finding the label apt innovation of our algorithm stands. In the standard A∗ strategy, the

search is guided to a single destination. In Dijkstra’s algorithm, all des- tinations are given the same priority. In order to guide the search to

a subset of primary destinations (the set *𝑇* of terminals), we select the

label *𝐿* = (*𝑢, 𝑑*(*𝐿*)*, 𝑡*(*𝐿*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*)) that minimizes value:

*𝑚𝑖𝑛*(*𝑑*(*𝐿*) + *𝑑𝑚𝑖𝑛*(*𝑢, 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣 , 𝑠*))

0

*𝑠*∈*𝑇*

We denote *𝐾*(*𝐿*) this value and call it the *key* of the label. For a given node *𝑠, 𝑑*(*𝐿*) + *𝑑𝑚𝑖𝑛*(*𝑢, 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*) is the minimum detour that could be achieved when extending label *𝐿* to *𝑠,* compared to the shortest path in distance between *𝑣*0 and *𝑠*. The key thus gives the minimal value among detours to all destinations in *𝑇 ,* that is, it prioritizes labels that could

potentially lead to one of the destinations effectively. Actually, with this key, the algorithm will first explore the min-distance paths leading to

nodes in *𝑇* (the key is zero for these paths), then it will progressively

deviate from these paths. Note that distances computed in preprocessing

are used when computing the key.

time paths have been generated for all terminals in *𝑇* . Indeed, The algorithm should terminate when the labels representing min-

any label with a larger detour would necessarily be dominated by these labels: both time and distance would be larger. More for-

mally, the min-time path between *𝑣*0 and *𝑠* is given by label *𝐿𝑠* =

(*𝑠, 𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*)*, 𝑡𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿𝑠* )). In *𝐿𝑠 ,* the detour is *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*) −

*𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*). Therefore, the stopping criterion is that the key of the selected

label is larger than max*𝑠*∈*𝑇 𝐾 𝑚𝑎𝑥* with *𝐾 𝑚𝑎𝑥* = *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*).

*𝑠 𝑠*

19: Compute path set P*𝑜𝑝𝑡* (*𝑣*0 *, 𝑠*) from all labels in *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑠*], for all *𝑠* ∈ *𝑇*

20: **return** P*𝑜𝑝𝑡* (*𝑣*0 *, 𝑠*) for all *𝑠* ∈ *𝑇*

The preprocessing is important to implement the A∗ mechanism. It works as follows:

and in time from node *𝑣*0 to all nodes *𝑣* ∈ *𝑉* . Four tables are con- *•* Using Dijkstra’s algorithm, we compute shortest paths in distance structed: *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*) and *𝑡𝑚𝑎𝑥*(*𝑣*0 *, 𝑣*) indicating distances and times as-

The selected label is given by function *Min()*, which indicates the label with minimal key, *i.e.*, the label that would be returned using function *extractMin()* (Line 4).

* 1. *Proof of optimality*

**Theorem 1.** *Algorithm MDA*∗ *provides minimal complete sets of efficient paths from source node 𝑣*0 *to all destination nodes 𝑠* ∈ *𝑇 .*

eﬃcient path *𝑃* from the source node *𝑣*0 to a destination node *𝑠* ∈ *𝑇* such **Proof.** Let us assume that at the end of [Algorithm 2](#_bookmark11) there exists an that (*𝑑*(*𝑃* )*, 𝑡*(*𝑃* )) does not belong to the set of non-dominated vector costs

returned by the algorithm. Under this assumption, the set of eﬃcient

**Theorem 2.** *The complexity of algorithm MDA*∗ *is in 𝑂*(*𝑚*Δ2 *𝑙𝑜𝑔*(*𝑛*Δ))*.*

**Proof.** Given a node *𝑣* ∈ *𝑉 ,* a label *𝐿* arriving at node *𝑣* has a chance to lead to an eﬃcient path at one of the destination nodes *𝑠* ∈ *𝑇* if *𝑑*(*𝐿*) +

*𝑑𝑚𝑖𝑛*(*𝑣, 𝑠*) ≤ *𝑑𝑚𝑎𝑥*(*𝑣 , 𝑠*). Therefore every label *𝐿* maintained at node *𝑣* dur-

paths between *𝑣*0 and *𝑠* is incomplete. We show that it cannot happen.

Let (*𝑢 , 𝑢 , ..., 𝑢* ) be the sequence of nodes visited along path *𝑃* with

0

ing the labeling procedure verifies *𝑑*(*𝐿*) ≤ *𝑚𝑎𝑥*(*𝑑*

*𝑚𝑎𝑥*

(*𝑣*0*, 𝑠*) − *𝑑*

*𝑚𝑖𝑛*

(*𝑣, 𝑠*)).

0 1 *𝑟*

*𝑠*∈*𝑇*

*𝑢*0 = *𝑣*0 and *𝑢𝑟* = *𝑠*. Let us denote by *𝑃*0*𝑖* and *𝑃𝑖𝑠* the paths respectively defined by node sequences (*𝑢*0 *, ..., 𝑢𝑖* ) and (*𝑢𝑖 , ..., 𝑢𝑟* ) for all *𝑖* ∈ {0*, ..., 𝑟*}. Let *𝐿𝑖* be the label associated with path *𝑃*0*𝑖* . Due to the Principle of

Optimality [Martins (1984b)](#_bookmark54), label *𝐿𝑖* can never be dominated by la-

In addition, *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*) ≤ *𝑑*(*𝐿*)*,* and, seeing that distances are assumed

the distance, the number of labels in list *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] is bounded by *𝛿*(*𝑣*). to be integer and that at most one label is kept for every value of

Equivalently, the total number of labels in heap *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠* is bounded by

bels in *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑢𝑖* ]. Consequently, the reason why *𝑃* is not found by

∑

*𝑣*∈*𝑉*

*𝛿*(*𝑣*) ≤ *𝑛*Δ.

[Algorithm 2](#_bookmark11) is that:

* either the algorithm stops before *𝑃* is generated, that is, there exists

*𝑗* ∈ {1*, ..., 𝑟* − 1} such that *𝐾* (*𝐿𝑗* ) *> 𝑚𝑎𝑥𝐾 𝑚𝑎𝑥*;

′ *𝑠*′

*𝑠* ∈*𝑇*

The complexities of the main procedures performed during the search are as follows:

* The extraction of the label with smallest key value requires

*𝑂*(*𝑙𝑜𝑔*(*𝑛*Δ)) operations;

* or there exists *𝑗* ∈ {1*, ..., 𝑟* − 1} such that a label *𝐿* ∈ *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑢𝑗* ] is

equal to *𝐿𝑗* .

In the second case, let *𝑃* ′ be the path obtained from *𝐿* by extending it with path *𝑃𝑗𝑠*. Seeing that (*𝑑*(*𝑃* ′)*, 𝑡*(*𝑃* ′)) = (*𝑑*(*𝑃* )*, 𝑡*(*𝑃* )) we fall into a situation similar to that of *𝑃* and, inductively, we eventually arrive to

the first case.

Without loss of generality, let us thus assume that label *𝐿𝑗* ob-

tained from path *𝑃* is such that *𝐾*(*𝐿* ) *> 𝑚𝑎𝑥𝐾 𝑚𝑎𝑥*. We know that *𝑑*(*𝑃* ) =

* + - For each new label *𝐿*′ = (*𝑣, 𝑑*(*𝐿*′)*, 𝑡*(*𝐿*′)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*′))*,* the dominance

in list *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] and the removal of dominated labels from *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] check which implies the insertion of the new (non-dominated) label and from the heap *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠,* requires *𝑂*(*𝛿*(*𝑣*)*𝑙𝑜𝑔*(*𝑛*Δ)) operations.

* + - The extension of a selected label *𝐿* = (*𝑢, 𝑑*(*𝐿*)*, 𝑡*(*𝐿*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*))

through all outgoing arcs (*𝑢, 𝑣*) ∈ *𝐴* requires *𝑂*(∑ (1 + *𝑙𝑜𝑔*(*𝑛*Δ) +

(*𝑢,𝑣*)∈*𝐴*

*𝛿*(*𝑣*)*𝑙𝑜𝑔*(*𝑛*Δ)) operations: the first term in the sum corresponds to the

generation of the new label, the second term corresponds to the in-

*𝑗 𝑠*′∈*𝑇 𝑠*′

*𝑑*(*𝐿𝑗* ) + *𝑑*(*𝑃𝑗𝑠*). From properties [1](#_bookmark5) and [2](#_bookmark6), we also know that *𝑑*(*𝑃* ) ≤

*𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*)*, i.e.*:

*𝑑*(*𝐿𝑗* ) + *𝑑*(*𝑃𝑗𝑠*) ≤ *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*)

sertion of the new label into the heap and the third term corresponds to the dominance check.

Since every label arriving at a node *𝑣* and that is in the

heap is selected once and *𝑂*(∑ (1 + *𝑙𝑜𝑔*(*𝑛*Δ) + *𝛿*(*𝑣*)*𝑙𝑜𝑔*(*𝑛*Δ))) ≤

Then,

∑ (*𝑢,𝑣*)∈*𝐴*

*𝑂*(

(*𝑢,𝑣*)∈*𝐴 𝛿*(*𝑣*)*𝑙𝑜𝑔*(*𝑛*Δ)*,* the total complexity of the algorithm is given by:

*𝑑*(*𝐿𝑗* ) + *𝑑𝑚𝑖𝑛*(*𝑢𝑗 , 𝑠*) ≤ *𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*)

complexity of MDA\*

and, so,

*𝑑*(*𝐿* ) + *𝑑𝑚𝑖𝑛*(*𝑢 , 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣 , 𝑠*) ≤ *𝑑𝑚𝑎𝑥*(*𝑣 , 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣 , 𝑠*)

(

∑

= *𝑂*

(

∑

*𝑙𝑜𝑔*(*𝑛*Δ) +

))

∑

*𝛿*(*𝑣*)*𝑙𝑜𝑔*(*𝑛*Δ)

*𝑗 𝑗* 0 0 0

*𝑢*∈*𝑉 𝐿*∈*𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑢*]

*𝑣*∈*𝑉* ;(*𝑢,𝑣*)∈*𝐴*

Seeing that *𝑑*(*𝐿* ) + *𝑑𝑚𝑖𝑛*(*𝑢 , 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣 , 𝑠*) = *𝐾*(*𝐿* ) and that *𝑑𝑚𝑎𝑥*(*𝑣 , 𝑠*) − ( ∑ ∑ )

*𝑗 𝑗*

0 *𝑗* 0

*𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*) = *𝐾 𝑚𝑎𝑥,* we obtain

*𝑠*

*𝐾*(*𝐿* ) ≤ *𝐾 𝑚𝑎𝑥* ≤ *𝑚𝑎𝑥𝐾 𝑚𝑎𝑥*

≤ *𝑂*

*𝑛*Δ*𝑙𝑜𝑔*(*𝑛*Δ) +

*𝐿*∈*𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑢*] (*𝑢,𝑣*)∈*𝐴*

( 2 )

Δ*𝑙𝑜𝑔*(*𝑛*Δ)

*𝑗 𝑠*

*𝑠*′ ∈*𝑇*

*𝑠*′

≤ *𝑂 𝑛*Δ*𝑙𝑜𝑔*(*𝑛*Δ) + *𝑚*Δ *𝑙𝑜𝑔*(*𝑛*Δ)

which contradicts our assumption and proves that the algorithm pro- vides complete sets of eﬃcient paths. In addition, these sets are minimal because the algorithm prevents from having two labels with the same vector cost attached to the same node. □

* 1. *Complexity analysis*

We analyze the complexity of algorithm MDA∗ with structures

*𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠* and *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] implemented as follows:

* + - Structure *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠* is a heap; it embeds the following methods:
      * *𝑀𝑖𝑛*() returns the label *𝐿* at the top of the heap, *i.e.*, with the smallest *𝐾*(*𝐿*) value; its complexity is in *𝑂*(1);
      * *𝑒𝑥𝑡𝑟𝑎𝑐𝑡𝑀𝑖𝑛*() extracts the label *𝐿* at the top of the heap; its com- plexity is in *𝑂*(*𝑙𝑜𝑔*(|*𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠*|));
      * *𝑎𝑑𝑑*(*𝐿*) inserts label *𝐿* into *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠*; its complexity is in

*𝑂*(*𝑙𝑜𝑔*(|*𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠*|));

* + - * *𝑟𝑒𝑚𝑜𝑣𝑒*(*𝐿*) removes label *𝐿* from *𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠*. its complexity is in

*𝑂*(|*𝑎𝑙𝑙𝐿𝑎𝑏𝑒𝑙𝑠*|).

* + - Structure *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*] is a chained list for each *𝑣* ∈ *𝑉* ; it gives access to

the following methods:

* + - * *𝑎𝑑𝑑*(*𝐿*) adds label *𝐿* to *𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*]; the complexity is in *𝑂*(1);
      * *𝑟𝑒𝑚𝑜𝑣𝑒*(*𝐿*) removes label *𝐿* from the list; it is performed in

*𝑂*(|*𝐿𝑎𝑏𝑒𝑙𝑠*[*𝑣*]|).

Without loss of generality, we assume that data are integer. If it is not the case, data can be multiplied by a power of ten. Let us introduce

≤ *𝑂*(*𝑚*Δ2 *𝑙𝑜𝑔*(*𝑛*Δ))

□

# Steiner BSPP with time windows

As already explained, this work is motivated by the computation of multigraph data in the context of vehicle routing problems. When two attributes (*e.g.*, time and distance) are present, it generally implies addi- tional constraints on routes. For example, in the VRPTW, time windows limit customer visit times. These constraints can be exploited in the com- putation, as we now show for the VRPTW.

Let *𝑒𝑠* and *𝑙𝑠* denote respectively the earliest starting service time and

the latest starting service time for a terminal *𝑠* ∈ *𝑇* . It is not allowed to reach customer *𝑠* after time *𝑙𝑠* ; arriving before *𝑒𝑠* is possible but implies waiting for the opening time *𝑒𝑠* . For the depot, this window corresponds

ing time. A path *𝑃* between two terminals *𝑢* and *𝑣* is feasible with respect to the time horizon: starting time from the depot, latest allowed return- to time windows if and only if *𝑒𝑢* + *𝑡*(*𝑃* ) ≤ *𝑙𝑣* . Therefore, sets of eﬃcient

paths can be limited to paths satisfying this condition.

node *𝑣*0 ∈ *𝑇* . We denote by *𝑇* +(*𝑣*0 ) the subset of terminals reachable from We consider the computation of eﬃcient paths starting from a given

*𝑣*0 within their time windows: *𝑇* +(*𝑣*0 ) = {*𝑠* ∈ *𝑇* ∶ *𝑒𝑣*0 + *𝑡𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*) ≤ *𝑙𝑠* } *⊂*

*𝑇* . Only destination nodes in *𝑇* +(*𝑣*0) should be considered during the

search procedure. To do this, we propose the following enhancements:

1. The key of a label *𝐿* = (*𝑢, 𝑑*(*𝐿*)*, 𝑡*(*𝐿*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿*)) is evaluated regarding

*𝛿*(*𝑣*) = *𝑚𝑎𝑥*(*𝑑𝑚𝑎𝑥*(*𝑣*0 *, 𝑠*) − *𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑣*) − *𝑑𝑚𝑖𝑛*(*𝑣, 𝑠*)) for every node *𝑣* ∈ *𝑉 ,* and

only reachable destination nodes: *𝐾*(*𝐿*) = *𝑚𝑖𝑛*

(*𝑑*(*𝐿*) + *𝑑𝑚𝑖𝑛*(*𝑢, 𝑠*) −

*𝑠*∈*𝑇*

let Δ = *𝑚𝑎𝑥*(*𝛿*(*𝑣*)).

*𝑣*∈*𝑉*

*𝑑𝑚𝑖𝑛*(*𝑣*0 *, 𝑠*));

*𝑠*∈*𝑇* + (*𝑣*0 )

Road network characteristics.

outgoing arcs

Number of instances for each road network.

number of terminals

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| network | n | m | min | max |  |  | network | *𝑛𝑇* = 26 | *𝑛𝑇* = 51 | *𝑛𝑇* = 101 | *𝑛𝑇* = 201 | *𝑛𝑇* = 501 |  |
| AIX-1 | 5437 | 10098 | 1 | 4 |  |  | AIX-1 | 10 | 10 | 10 | 10 | 10 |  |
| AIX-2 | 19500 | 36203 | 1 | 5 |  |  | AIX-2 | 10 | 10 | 10 | 10 | 10 |  |
| DC | 9559 | 29707 | 1 | 6 |  |  | DC | 10 | 10 | 10 | 10 | 10 |  |
| DE | 49109 | 119838 | 1 | 6 |  |  | DE | 10 | 10 | 10 | 10 | 10 |  |
| RI | 53658 | 137596 | 1 | 6 |  |  | RI | 10 | 10 | 10 | 10 | 10 |  |
| AK | 69082 | 155245 | 1 | 6 |  |  | AK | 10 | 10 | 10 | 10 | 10 |  |

1. The algorithm should terminate once labels *𝐿𝑠* = (*𝑠, 𝑑𝑚𝑎𝑥*(*𝑣*0*, 𝑠*)*, 𝑡𝑚𝑖𝑛*(*𝑣*0*, 𝑠*)*, 𝑓𝑎𝑡ℎ𝑒𝑟*(*𝐿𝑠*)) have been generated for destination nodes *𝑠* ∈ *𝑇* +(*𝑣*0). Thus, the stopping criterion is that

nodes *𝑛* and the number of arcs *𝑚,* respectively. The last two columns

give the minimum and the maximum number of outgoing arcs over all

nodes in the road network.

the selected label key is larger than *𝑚𝑎𝑥*

*𝐾𝑚𝑎𝑥*;

From each road network, we generated 50 instances by randomly

*𝑠*∈*𝑇* +(*𝑣*0 ) *𝑠*

path should be considered. Every node *𝑢* such that *𝑒𝑣*0 + *𝑡𝑚𝑖𝑛*(*𝑣*0*, 𝑢*) + 3. Only nodes that are apt to lead to a destination node with a feasible

*𝑡𝑚𝑖𝑛*(*𝑢, 𝑠*) *> 𝑙𝑠* for all destination nodes *𝑠* ∈ *𝑇* +(*𝑣*0) is discarded from

graph *𝐺*.

Due to the first enhancement, label keys increase more quickly dur- ing the labeling procedure. The second enhancement tightens the stop- ping condition. Combined with the first enhancement, it limits the num- ber of labels generated during the search. The third enhancement is per- formed in preprocessing and permits reducing the size of the graph.

# Computational experiments

In this section, we present the computational experiments carried out to evaluate the eﬃciency of the proposed solution method. First, we present the benchmark problems used in the experiments. Then, we re- port the computational results and we analyse the impact of considering the time windows on the algorithm performance.

All algorithms are implemented in the C++ programming language and tests are run on an Intel Xeon(R) CPU E5-2620v2 2.1 GHz computer with 32GB of memory.

* 1. *Test problems*

Since we are interested in computing paths for transportation prob- lems, we conducted our computational experiments on the basis of two series of real-world road networks:

* + - Two road networks (AIX-1 and AIX-2) were constructed based on spatial data from the city of *Aix-en-Provence*[1](#_bookmark17) in France provided by **OpenStreetMap**⋇c [2](#_bookmark18) database. Each road segment is defined by a length, a maximum allowed speed and a travel direction. Travel times were then computed using speeds and lengths.
    - Four road networks (DC, DE, RI and AK) were extracted by [Schultes (2005)](#_bookmark57) from **US Census**[3](#_bookmark19) and correspond to *Washington D.C., Delaware, Rhode Island* and *Alaska*, in the United States, respec- tively. In these networks, each road segment is given with a distance and a travel time. Note that, the road networks in the original data are undirected and we converted them into directed networks by duplicating all arcs.

[Table 1](#_bookmark15) reports the main characteristics of these 6 road networks. For each road network, the first two columns indicate the number of

1 Aix-en-Provence is a city-commune in the region of *Provence-Alpes-Cote* *d’Azur* in the south of France, about 30 km north of *Marseilles*

2 OpenStreetMap is a collaborative project wich creates and distributes freely

available geospatial data. [www.openstreetmap.org/](http://www.openstreetmap.org/)

3 US Census 2000 TIGER/Line Files. U.S. Census Bureau, Washing- [ton, DC, Geography Division. http://www.census.gov/geo/www/tiger/ tigerua/uatgr2k.html](http://www/tiger/tigerua/uatgr2k.html)

selecting terminals among the network nodes. We considered different terminal set sizes and two different configurations:

* + - Random: terminals are randomly drawn on the whole road network graph.
    - Centered: all terminals but one are randomly selected in the center of the network (a fourth of the area covered by the network, centered). The remaining terminal is randomly selected near the south-west corner of the network. It represents the standard situation of urban deliveries from a distant depot.

each value of *𝑛𝑇* . For each configuration, 10 instances are generated: 5 [Table 2](#_bookmark15) details the number of instances for each road network and for

random instances and 5 centered instances. It amounts to a total of 300 instances.

* 1. *Computational results*

its results to those of three state-of-the-art algorithms: A∗, LSET and In order to evaluate the performance of our algorithm, we compare

LCOR.

* + - A∗ follows the classical guided-search mechanism. The core algo-

origin to a single destination. Contrary to MDA∗, the key of a label is rithm solves the bi-objective shortest path problem from a single

given by the minimal distance to reach the destination. It is repeated for all pairs of terminals.

* + - LSET is a one-to-all label setting algorithm. The search is not guided. At each iteration, the minimum label according to the lexicographic

order is selected. The stopping criterion defined for MDA∗ is used.

The algorithm is repeated for all possible origin points in *𝑇* .

* + - LCOR is a one-to-all label correcting algorithm. The search is not

guided either. At each iteration, a node is selected and the labels associated with this node are extended to successor nodes. Nodes

repeated for all possible origin points in *𝑇* . reenter the queue when their label list is modified. The algorithm is

For each instance, MDA∗ is also applied *𝑛𝑇* times, once for each node in *𝑇* selected as the source node, so that complete sets of eﬃcient paths

are obtained between all pairs of terminals.

Results for random instances are presented in [Tables 3](#_bookmark20), [5](#_bookmark20), [7](#_bookmark22), [9](#_bookmark22), [11](#_bookmark24) and [13](#_bookmark25), for road networks AIX-1, AIX-2, DC, DE, RI and AK, respectively. Re- sults for centered instances are presented in [Tables 4, 6](#_bookmark21), [8,10](#_bookmark23), [12](#_bookmark26) and [14](#_bookmark27), for road networks AIX-1, AIX-2, DC, DE, RI and AK, respectively.

Columns “MDA∗”, “A∗”, “LSET” and “LCOR” report total computing

times (in seconds) for the four algorithms. The average number of eﬃ-

cient paths between pairs of terminals is presented in column “#p”.

A first observation in these tables is that, apart from a few exceptions, results are homogeneous among the 5 instance replications for a given

value *𝑛𝑇* and a given configuration (random or centered).

Comparing MDA∗ and A∗, the latter is consistently beaten. A∗ is only competitive for small values of *𝑛𝑇 ,* when the number of executions of

Computing times for AIX-1 - Random. Computing times for AIX-2 - Random.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |  | *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |
| 26 | 1 | 0.5 | 1.0 | 0.5 | 1.4 | 3.8 |  | 26 | 1 | 9.3 | 24.5 | 23.8 | 169.7 | 15.6 |
|  | 2 | 0.6 | 1.2 | 0.6 | 1.6 | 4.3 |  |  | 2 | 7.4 | 19.0 | 20.7 | 134.3 | 14.3 |
|  | 3 | 0.5 | 1.1 | 0.6 | 1.4 | 4.1 |  |  | 3 | 9.7 | 25.6 | 20.7 | 120.6 | 13.6 |
|  | 4 | 0.5 | 1.0 | 0.6 | 1.4 | 4.1 |  |  | 4 | 9.1 | 27.7 | 22.7 | 152.8 | 16.1 |
|  | 5 | 0.5 | 1.2 | 0.5 | 1.4 | 4.0 |  |  | 5 | 7.7 | 28.7 | 27.1 | 156.1 | 15.9 |
| 51 | 1 | 1.2 | 3.7 | 1.1 | 2.9 | 3.9 |  | 51 | 1 | 19.9 | 80.8 | 43.3 | 296.2 | 14.5 |
|  | 2 | 1.1 | 3.3 | 1.1 | 2.8 | 4.0 |  |  | 2 | 21.2 | 91.3 | 41.5 | 276.4 | 14.8 |
|  | 3 | 1.1 | 3.5 | 1.0 | 2.8 | 4.1 |  |  | 3 | 16.6 | 77.3 | 38.5 | 245.5 | 13.3 |
|  | 4 | 1.2 | 4.0 | 1.1 | 2.8 | 4.3 |  |  | 4 | 23.7 | 91.3 | 45.5 | 298.4 | 14.7 |
|  | 5 | 1.2 | 4.0 | 1.1 | 3.1 | 4.2 |  |  | 5 | 23.7 | 113.5 | 45.2 | 342.0 | 15.2 |
| 101 | 1 | 2.6 | 11.5 | 2.2 | 5.4 | 4.1 |  | 101 | 1 | 47.6 | 328.5 | 81.3 | 561.8 | 14.5 |
|  | 2 | 2.5 | 10.8 | 2.1 | 5.2 | 3.9 |  |  | 2 | 48.6 | 345.0 | 81.5 | 551.6 | 14.4 |
|  | 3 | 2.6 | 13.4 | 2.3 | 5.8 | 4.1 |  |  | 3 | 45.5 | 319.5 | 79.1 | 534.1 | 13.3 |
|  | 4 | 2.6 | 12.9 | 2.1 | 5.2 | 4.0 |  |  | 4 | 36.0 | 229.2 | 70.8 | 509.4 | 12.6 |
|  | 5 | 2.5 | 11.8 | 2.2 | 5.3 | 4.1 |  |  | 5 | 45.9 | 344.0 | 78.4 | 615.3 | 14.0 |
| 201 | 1 | 6.0 | 43.4 | 4.3 | 10.6 | 4.1 |  | 201 | 1 | 105.3 | 1323.6 | 162.8 | 1106.8 | 14.4 |
|  | 2 | 6.3 | 51.8 | 4.4 | 11.0 | 4.1 |  |  | 2 | 94.0 | 1062.8 | 149.1 | 1041.2 | 13.0 |
|  | 3 | 6.2 | 47.2 | 4.3 | 10.6 | 4.1 |  |  | 3 | 100.7 | 1273.8 | 153.9 | 1170.7 | 13.9 |
|  | 4 | 6.4 | 54.1 | 4.4 | 11.5 | 4.2 |  |  | 4 | 87.0 | 960.4 | 143.1 | 1000.1 | 12.9 |
|  | 5 | 6.3 | 51.8 | 4.2 | 11.2 | 4.2 |  |  | 5 | 96.6 | 1069.8 | 163.0 | 1034.5 | 12.6 |
| 501 | 1 | 22.5 | 285.0 | 11.4 | 27.0 | 4.1 |  | 501 | 1 | 285.5 | 7365.8 | 375.2 | 2767.1 | 13.8 |
|  | 2 | 22.9 | 278.1 | 10.6 | 28.0 | 4.2 |  |  | 2 | 274.1 | 5831.1 | 384.4 | 2637.6 | 12.9 |
|  | 3 | 22.8 | 263.6 | 12.4 | 29.1 | 4.2 |  |  | 3 | 269.6 | 6200.6 | 399.8 | 2597.3 | 13.0 |
|  | 4 | 22.6 | 255.7 | 12.4 | 27.1 | 4.2 |  |  | 4 | 273.7 | 6405.3 | 399.4 | 2587.4 | 13.3 |
|  | 5 | 23.0 | 275.0 | 12.8 | 30.7 | 4.4 |  |  | 5 | 275.6 | 6664.8 | 395.1 | 2697.6 | 13.6 |
| **Table 6**  **Table 4** Computing times for AIX-2 - Centered. | | | | | | | | | | | | | | |

Computing times for AIX-1 - Centered.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |  | 26 | 1 | 2.7 | 5.4 | 12.8 | 106.7 | 8.2 |
| 26 | 1 | 0.4 | 0.5 | 0.5 | 0.9 | 2.6 |  |  | 2 | 3.5 | 7.6 | 14.9 | 118.0 | 10.6 |
|  | 2 | 0.4 | 0.6 | 0.5 | 0.9 | 3.0 |  |  | 3 | 2.6 | 5.3 | 10.4 | 66.1 | 7.2 |
|  | 3 | 0.4 | 0.5 | 0.5 | 1.0 | 3.1 |  |  | 4 | 3.3 | 6.6 | 14.2 | 95.1 | 8.9 |
|  | 4 | 0.4 | 0.6 | 0.5 | 1.0 | 3.0 |  |  | 5 | 2.8 | 5.5 | 10.6 | 75.9 | 8.2 |
|  | 5 | 0.4 | 0.6 | 0.6 | 1.3 | 3.5 |  | 51 | 1 | 7.2 | 21.1 | 24.4 | 239.7 | 8.6 |
| 51 | 1 | 0.8 | 1.3 | 0.8 | 1.7 | 2.6 |  |  | 2 | 7.7 | 25.1 | 28.9 | 225.1 | 10.0 |
|  | 2 | 0.9 | 1.5 | 1.0 | 2.2 | 3.1 |  |  | 3 | 6.5 | 20.6 | 21.0 | 144.3 | 7.7 |
|  | 3 | 0.8 | 1.5 | 1.1 | 2.3 | 3.2 |  |  | 4 | 7.4 | 22.1 | 25.0 | 181.5 | 8.7 |
|  | 4 | 0.8 | 1.4 | 1.0 | 2.0 | 3.0 |  |  | 5 | 6.5 | 20.0 | 25.6 | 165.2 | 8.3 |
|  | 5 | 0.8 | 1.4 | 0.9 | 2.1 | 2.8 |  | 101 | 1 | 16.0 | 83.4 | 52.2 | 439.7 | 8.8 |
| 101 | 1 | 1.9 | 3.9 | 1.6 | 3.5 | 2.8 |  |  | 2 | 17.5 | 98.5 | 53.0 | 432.4 | 9.7 |
|  | 2 | 1.8 | 4.0 | 1.7 | 3.7 | 2.7 |  |  | 3 | 15.0 | 80.2 | 43.6 | 292.7 | 7.6 |
|  | 3 | 1.9 | 4.6 | 2.2 | 4.7 | 3.1 |  |  | 4 | 16.8 | 81.2 | 46.3 | 369.0 | 8.9 |
|  | 4 | 1.9 | 4.4 | 1.8 | 4.0 | 3.0 |  |  | 5 | 17.6 | 86.0 | 50.9 | 350.8 | 8.5 |
|  | 5 | 1.9 | 4.3 | 1.8 | 4.0 | 2.8 |  | 201 | 1 | 35.1 | 295.6 | 100.5 | 824.6 | 8.4 |
| 201 | 1 | 4.6 | 14.2 | 3.5 | 7.7 | 2.8 |  |  | 2 | 37.7 | 368.2 | 98.8 | 818.7 | 9.1 |
|  | 2 | 4.7 | 12.8 | 3.4 | 7.2 | 2.7 |  |  | 3 | 36.7 | 300.6 | 94.2 | 655.9 | 8.1 |
|  | 3 | 4.7 | 15.8 | 3.5 | 8.1 | 3.0 |  |  | 4 | 38.2 | 341.9 | 99.3 | 792.3 | 8.8 |
|  | 4 | 4.6 | 14.8 | 3.1 | 7.5 | 2.9 |  |  | 5 | 38.5 | 332.4 | 96.9 | 708.0 | 8.5 |
|  | 5 | 4.6 | 14.3 | 3.3 | 7.2 | 2.9 |  | 501 | 1 | 123.1 | 1689.8 | 255.5 | 1934.3 | 8.4 |
| 501 | 1 | 18.2 | 76.8 | 8.5 | 19.0 | 2.9 |  |  | 2 | 120.7 | 1821.6 | 257.2 | 1918.7 | 8.8 |
|  | 2 | 18.0 | 79.2 | 8.7 | 18.7 | 2.8 |  |  | 3 | 123.0 | 1809.6 | 244.1 | 1808.1 | 8.4 |
|  | 3 | 18.1 | 77.7 | 7.7 | 19.2 | 2.8 |  |  | 4 | 125.8 | 1560.3 | 238.5 | 1888.2 | 8.1 |
|  | 4 | 18.2 | 73.9 | 9.0 | 19.2 | 2.8 |  |  | 5 | 124.3 | 1838.9 | 250.2 | 2039.8 | 8.7 |
|  | 5 | 18.0 | 78.1 | 8.6 | 20.0 | 2.8 |  |  |  |  |  |  |  |  |

*𝑛𝑇* MDA∗ A∗ LSET LCOR #p

the one-to-one core algorithm remains limited. Clearly, the advantage

of *𝑛𝑇* increases and reaches values that are practically meaningful in the of driving the search to a single direction is quickly lost when the size

context of vehicle routing.

Similarly, except for a few exceptions, LSET is more eﬃcient than

of the analysis we focus on MDA∗ and LSET. Except for the smaller graph LCOR, sometimes very significantly. For that reason, in the remainder (AIX-1), MDA∗ is always many times faster than LSET, often up to 10 times. Not surprisingly, MDA∗ is specially interesting when *𝑛𝑇* is small.

In this case indeed, being able to limit the search to some promising directions really makes a difference, compared to an algorithm that in-

smaller *𝑛𝑇 ,* the larger the benefits achieved with MDA∗. differently explores all directions. Globally, the larger the graph and the

Result tables for Centered instances even strengthen these observa- tions. Let us recall that this setting is particularly relevant in practice, as it typically corresponds to the situation encountered by urban logis-

customers is indeed particularly beneficial for MDA∗. When the source tics providers. Having a distant depot (south-west corner) with centered node (*𝑣*0) is the depot, all destination nodes are clustered and the search

is very eﬃciently guided. When the source node is any other terminal, all destination nodes but one are close; the best paths to these nodes

also eﬃciently computed thanks to the A∗ mechanism. On the contrary, can eﬃciently be found; the best paths to the last node (the depot) are

LSET always needs to compute best paths to a large part of the network before being able to stop the search.

[Tables 3](#_bookmark20) to [14](#_bookmark27) also give interesting insights on the size of complete

this value only slightly depends on value *𝑛𝑇 ,* as could be expected, but sets of eﬃcient paths in the different networks. For a given network,

Computing times for DC - Random. Computing times for DE - Random.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |  | *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |
| 26 | 1 | 5.5 | 13.3 | 22.4 | 144.7 | 15.1 |  | 26 | 1 | 41.9 | 89.1 | 2436.2 | 1845.4 | 20.8 |
|  | 2 | 6.4 | 21.1 | 25.7 | 131.6 | 15.2 |  |  | 2 | 25.4 | 65.3 | 2003.1 | 1294.0 | 14.6 |
|  | 3 | 5.9 | 18.7 | 21.0 | 112.2 | 13.2 |  |  | 3 | 60.8 | 227.1 | 2338.6 | 1914.2 | 20.9 |
|  | 4 | 8.0 | 27.4 | 27.3 | 183.9 | 19.1 |  |  | 4 | 38.3 | 100.0 | 2319.0 | 2426.0 | 21.8 |
|  | 5 | 5.6 | 15.6 | 21.1 | 137.3 | 15.5 |  |  | 5 | 49.7 | 126.8 | 2091.5 | 1693.1 | 16.3 |
| 51 | 1 | 14.7 | 63.8 | 48.4 | 272.8 | 14.7 |  | 51 | 1 | 83.7 | 275.1 | 4350.8 | 2978.5 | 16.9 |
|  | 2 | 16.1 | 87.3 | 48.6 | 287.0 | 15.7 |  |  | 2 | 145.1 | 677.3 | 4702.5 | 4464.7 | 22.5 |
|  | 3 | 15.3 | 64.7 | 46.3 | 301.4 | 16.4 |  |  | 3 | 175.1 | 647.8 | 4397.2 | 5331.1 | 22.9 |
|  | 4 | 13.0 | 58.6 | 41.0 | 263.9 | 14.3 |  |  | 4 | 136.8 | 565.4 | 5353.1 | 4093.4 | 19.4 |
|  | 5 | 16.5 | 80.1 | 47.4 | 345.4 | 17.3 |  |  | 5 | 122.0 | 428.1 | 4609.7 | 3976.9 | 19.0 |
| 101 | 1 | 34.6 | 300.0 | 96.7 | 556.5 | 15.2 |  | 101 | 1 | 280.2 | 1795.0 | 8993.5 | 7332.3 | 19.3 |
|  | 2 | 32.2 | 241.9 | 87.4 | 555.5 | 15.1 |  |  | 2 | 391.5 | 2536.7 | 9621.7 | 9492.8 | 21.3 |
|  | 3 | 42.1 | 397.8 | 119.4 | 719.2 | 18.7 |  |  | 3 | 284.9 | 1656.3 | 9547.9 | 7856.1 | 18.7 |
|  | 4 | 37.8 | 295.5 | 86.5 | 686.8 | 17.3 |  |  | 4 | 283.3 | 1511.6 | 9235.2 | 8017.7 | 18.5 |
|  | 5 | 37.7 | 358.6 | 88.5 | 589.4 | 15.3 |  |  | 5 | 299.5 | 1695.9 | 9120.2 | 7324.3 | 17.2 |
| 201 | 1 | 72.6 | 1064.9 | 185.9 | 1109.4 | 15.3 |  | 201 | 1 | 797.8 | 8648.8 | 18258.2 | 16861.5 | 20.2 |
|  | 2 | 84.5 | 1334.4 | 215.8 | 1377.4 | 17.7 |  |  | 2 | 671.5 | 6257.1 | 18681.4 | 15227.2 | 18.5 |
|  | 3 | 78.0 | 1359.5 | 188.3 | 1170.9 | 16.0 |  |  | 3 | 640.2 | 5823.0 | 18601.4 | 14473.7 | 16.5 |
|  | 4 | 79.9 | 1281.1 | 185.4 | 1257.4 | 16.5 |  |  | 4 | 722.6 | 7472.9 | 21660.4 | 16124.0 | 19.0 |
|  | 5 | 73.9 | 1129.1 | 210.0 | 1154.7 | 15.6 |  |  | 5 | 743.2 | 6655.7 | 17283.4 | 15907.9 | 18.2 |
| 501 | 1 | 211.0 | 7961.5 | 527.2 | 3071.1 | 16.3 |  | 501 | 1 | 2148.6 | 48383.4 | 45874.4 | 39798.4 | 19.0 |
|  | 2 | 205.2 | 7722.3 | 539.0 | 2825.3 | 16.1 |  |  | 2 | 2086.3 | 42608.7 | 47167.5 | 39141.1 | 18.0 |
|  | 3 | 201.6 | 7444.0 | 520.5 | 2856.1 | 15.4 |  |  | 3 | 2075.5 | 38102.2 | 52560.1 | 39243.8 | 18.5 |
|  | 4 | 205.0 | 7602.0 | 485.4 | 3014.1 | 15.9 |  |  | 4 | 2108.1 | 45133.2 | 49693.9 | 37259.4 | 18.6 |
|  | 5 | 199.6 | 7307.8 | 542.7 | 2771.6 | 15.3 |  |  | 5 | 2404.2 | 51901.8 | 49771.4 | 43776.5 | 20.7 |

**Table 8**

Computing times for DC - Centered.

**Table 10**

Computing times for DE - Centered.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |  | *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |
| 26 | 1 | 4.1 | 11.1 | 19.3 | 140.4 | 14.4 |  | 26 | 1 | 5.3 | 6.6 | 923.5 | 1038.2 | 5.2 |
|  | 2 | 3.5 | 7.9 | 27.5 | 143.8 | 12.6 |  |  | 2 | 6.0 | 8.7 | 854.2 | 967.0 | 6.6 |
|  | 3 | 3.7 | 10.3 | 22.1 | 145.6 | 13.9 |  |  | 3 | 5.2 | 6.6 | 983.4 | 915.5 | 5.3 |
|  | 4 | 2.7 | 6.2 | 15.7 | 163.2 | 13.2 |  |  | 4 | 5.8 | 7.8 | 790.3 | 1130.4 | 5.9 |
|  | 5 | 2.9 | 6.3 | 19.1 | 132.3 | 13.1 |  |  | 5 | 5.4 | 7.9 | 1239.0 | 1305.5 | 6.4 |
| 51 | 1 | 8.3 | 37.1 | 43.1 | 292.6 | 14.2 |  | 51 | 1 | 10.8 | 16.2 | 1710.3 | 1860.8 | 5.2 |
|  | 2 | 9.3 | 46.9 | 48.3 | 326.5 | 15.4 |  |  | 2 | 11.1 | 18.8 | 1769.1 | 2296.3 | 5.5 |
|  | 3 | 8.1 | 36.6 | 49.7 | 306.8 | 14.6 |  |  | 3 | 12.2 | 24.5 | 2295.2 | 2460.0 | 6.8 |
|  | 4 | 8.0 | 36.4 | 48.6 | 284.0 | 14.0 |  |  | 4 | 11.2 | 17.5 | 1799.8 | 2217.7 | 6.0 |
|  | 5 | 8.7 | 42.0 | 44.1 | 288.3 | 14.4 |  |  | 5 | 11.5 | 19.4 | 2070.7 | 1925.9 | 6.0 |
| 101 | 1 | 17.0 | 122.7 | 83.9 | 565.9 | 13.6 |  | 101 | 1 | 23.6 | 51.4 | 3293.0 | 4086.3 | 5.3 |
|  | 2 | 20.1 | 170.0 | 92.2 | 659.9 | 15.7 |  |  | 2 | 25.3 | 65.9 | 3865.4 | 4740.1 | 6.4 |
|  | 3 | 19.4 | 168.4 | 110.2 | 658.5 | 15.3 |  |  | 3 | 24.7 | 60.8 | 3732.4 | 4201.3 | 6.1 |
|  | 4 | 17.9 | 147.1 | 92.0 | 590.1 | 14.3 |  |  | 4 | 23.3 | 55.8 | 3983.9 | 4000.4 | 5.5 |
|  | 5 | 20.4 | 180.4 | 93.6 | 631.8 | 15.4 |  |  | 5 | 22.2 | 52.0 | 3743.0 | 4566.0 | 5.4 |
| 201 | 1 | 40.7 | 589.5 | 218.1 | 1119.7 | 14.0 |  | 201 | 1 | 54.6 | 187.6 | 6662.8 | 8491.9 | 5.6 |
|  | 2 | 41.6 | 570.6 | 207.3 | 1223.6 | 14.6 |  |  | 2 | 55.5 | 200.7 | 7284.9 | 8216.5 | 5.8 |
|  | 3 | 41.9 | 603.6 | 174.4 | 1260.4 | 14.7 |  |  | 3 | 54.0 | 200.5 | 7479.9 | 8329.5 | 5.9 |
|  | 4 | 37.7 | 508.1 | 166.3 | 1160.8 | 13.6 |  |  | 4 | 58.7 | 218.9 | 7469.4 | 9338.0 | 6.4 |
|  | 5 | 35.9 | 447.6 | 197.4 | 1130.0 | 13.2 |  |  | 5 | 57.9 | 261.6 | 7929.7 | 8876.5 | 6.4 |
| 501 | 1 | 117.9 | 3567.4 | 618.0 | 2866.3 | 14.1 |  | 501 | 1 | 189.4 | 1112.5 | 16953.1 | 21155.7 | 5.7 |
|  | 2 | 115.1 | 3443.5 | 434.3 | 3001.9 | 14.1 |  |  | 2 | 187.1 | 1333.4 | 19643.9 | 21895.6 | 6.3 |
|  | 3 | 111.7 | 3320.5 | 525.6 | 2911.0 | 13.8 |  |  | 3 | 188.9 | 1295.8 | 17553.3 | 21035.5 | 6.0 |
|  | 4 | 114.2 | 3368.3 | 413.2 | 2931.8 | 14.1 |  |  | 4 | 188.7 | 1324.7 | 17387.9 | 21497.7 | 6.2 |
|  | 5 | 115.4 | 3158.2 | 446.1 | 2949.8 | 13.7 |  |  | 5 | 200.5 | 1305.3 | 18159.4 | 21075.8 | 6.0 |

is different for the Random and Centered configurations. In the Cen- tered configuration, the average distance between terminals is smaller and one could expect a more limited number of eﬃcient paths. This is confirmed by experiments, but, surprisingly, the reduction factor is ex- tremely dependent on the network (almost null for DC, very large for DE or AK). Anyway, one can see that the average size of eﬃcient sets approximately varies between 3 and 40, which can be interpreted in two different manners. On the one hand, it remains very limited compared to the theoretical exponential-size that these sets could have. On the other hand, time and distance are strongly correlated and it is generally ad- mitted that the longer the distance, the longer the traveling time. Values of #p show that it is not completely true and that the number of eﬃ- cient paths between two terminals in a road-network can be relatively important.

* 1. *Results in the context of the VRPTW*

We now evaluate the impact of the enhancements proposed in [Section 5](#_bookmark13) when time windows are introduced. For the sake of brevity, we limit our experiments to the 25 AIX-2 instances and the 25 DC instances, with nodes randomly drawn. Time windows are added as follows:

1. A node, denoted *𝑠*0*,* is randomly selected in *𝑇* to represent the depot;

other terminals represent customers.

1. The depot time window [0*, 𝑙*0] defines the time horizon and is set such that every node in *𝑇* ⧵ {*𝑠*0} can be visited on a back-and-forth

route.

1. A feasible VRP solution is then constructed with a simple greedy algorithm.

**Table 11**

Computing times for RI - Random.

*𝑛𝑇* MDA∗ A∗ LSET LCOR #p

**Table 13**

Computing times for AK - Random.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 26 | 1 | 247.6 | 809.6 | 1711.0 | 12453.8 | 35.1 |  | *𝑛𝑇* |  | MDA∗ | A∗ | LSET | LCOR | #p |
|  | 2 | 233.3 | 685.1 | 1841.4 | 11315.9 | 31.7 |  | 26 | 1 | 217.6 | 733.1 | 1111.6 | 2731.9 | 34.2 |
|  | 3 | 361.6 | 1197.9 | 2099.0 | 15145.3 | 40.0 |  |  | 2 | 164.0 | 626.8 | 1127.4 | 2303.3 | 35.8 |
|  | 4 | 235.1 | 725.7 | 2327.8 | 13142.7 | 37.3 |  |  | 3 | 292.1 | 1050.0 | 1309.6 | 3037.7 | 37.3 |
|  | 5 | 758.2 | 2566.0 | 3693.9 | 28150.2 | 57.9 |  |  | 4 | 161.1 | 633.4 | 1053.8 | 2665.7 | 35.9 |
| 51 | 1 | 596.0 | 2674.4 | 3433.6 | 23524.0 | 32.6 |  |  | 5 | 260.8 | 1553.4 | 1456.1 | 3147.2 | 45.8 |
|  | 2 | 758.7 | 3905.8 | 4168.2 | 26257.5 | 38.0 |  | 51 | 1 | 423.5 | 3077.9 | 2184.7 | 4971.3 | 36.0 |
|  | 3 | 1166.0 | 8649.9 | 5739.0 | 41957.7 | 49.4 |  |  | 2 | 539.5 | 3389.7 | 2322.9 | 5600.6 | 39.2 |
|  | 4 | 556.5 | 2763.9 | 3448.7 | 22908.7 | 34.2 |  |  | 3 | 448.7 | 3295.3 | 2307.0 | 5347.0 | 38.2 |
|  | 5 | 711.1 | 3796.6 | 4530.1 | 27262.1 | 34.7 |  |  | 4 | 402.5 | 2502.3 | 2008.4 | 4635.5 | 33.7 |
| 101 | 1 | 1513.8 | 13173.9 | 7454.2 | 49607.5 | 35.3 |  |  | 5 | 460.5 | 3274.5 | 2001.6 | 5273.7 | 37.2 |
|  | 2 | 1943.1 | 24201.7 | 9107.0 | 64238.5 | 42.3 |  | 101 | 1 | 1109.6 | 12536.9 | 4454.1 | 10095.5 | 37.0 |
|  | 3 | 1504.2 | 14482.2 | 8485.1 | 51957.0 | 35.7 |  |  | 2 | 960.3 | 11787.0 | 4285.7 | 10050.4 | 36.0 |
|  | 4 | 1645.6 | 18710.7 | 8534.4 | 54202.1 | 38.4 |  |  | 3 | 945.5 | 9965.8 | 3788.1 | 9376.4 | 33.7 |
|  | 5 | 1073.2 | 8629.2 | 8233.8 | 46816.5 | 33.1 |  |  | 4 | 1151.9 | 14056.3 | 4720.7 | 10744.0 | 37.5 |
| 201 | 1 | 3105.8 | 54667.5 | 16072.7 | 100117.0 | 36.0 |  |  | 5 | 1057.6 | 9883.7 | 4473.6 | 10041.0 | 34.5 |
|  | 2 | 3020.0 | 51102.8 | 15736.8 | 96977.3 | 33.8 |  | 201 | 1 | 2396.4 | 49467.7 | 8609.2 | 20472.4 | 37.1 |
|  | 3 | 3125.2 | 53022.5 | 15586.0 | 101538.0 | 36.2 |  |  | 2 | 2355.0 | 46347.2 | 8381.8 | 19343.5 | 35.3 |
|  | 4 | 3481.9 | 70638.8 | 18784.7 | 111291.0 | 36.7 |  |  | 3 | 2503.1 | 47041.6 | 9202.7 | 21304.7 | 37.0 |
|  | 5 | 3732.5 | 77161.3 | 18149.0 | 116441.0 | 39.5 |  |  | 4 | 2499.4 | 57734.6 | 8992.6 | 22188.9 | 39.8 |
| 501 | 1 | 8220.0 | 333443.0 | 39876.1 | 244181.0 | 34.9 |  |  | 5 | 2338.8 | 54599.6 | 9636.2 | 20946.5 | 37.4 |
|  | 2 | 9371.8 | 36338.7 | 46338.7 | 276690.0 | 37.9 |  | 501 | 1 | 6584.5 | 291573.0 | 21734.9 | 50337.2 | 35.8 |
|  | 3 | 8154.8 | 308090.5 | 40890.9 | 26892.7 | 34.1 |  |  | 2 | 6837.0 | 349924.0 | 25313.8 | 54515.3 | 38.6 |
|  | 4 | 9027.8 | 328953.4 | 42953.3 | 27954.1 | 37.5 |  |  | 3 | 6597.9 | 300878.0 | 20646.0 | 50640.3 | 36.2 |
|  | 5 | 9250.2 | 320034.0 | 40035.0 | 25021.1 | 37.5 |  |  | 4 | 6316.4 | 305386.0 | 21351.3 | 50216.9 | 36.2 |
|  |  |  |  |  |  |  |  |  | 5 | 7058.7 | 337687.0 | 23276.6 | 53547.0 | 38.2 |

**Table 12**

Computing times for RI - Centered.

*𝑛𝑇* MDA∗ A∗ LSET LCOR #p

26 1 148.7 660.6 1996.1 12324.5 27.0

**Table 14**

Computing times for AK - Centered.

2 111.9 409.1 1699.6 11821.7 29.4

3 87.9 218.8 1464.7 7863.0 18.6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4 | 98.2 | 375.1 | 1815.6 | 8796.0 | 20.7 |  | 26 | 1 | 109.4 | 138.5 | 1425.1 | 2563.6 | 12.6 |
| 5 | 58.1 | 157.7 | 1378.8 | 9121.0 | 20.3 |  |  | 2 | 13.8 | 19.2 | 1357.8 | 2291.2 | 9.5 |
| 51 | 1 | 269.3 | 1657.9 | 3376.6 | 21111.2 | 24.6 |  |  | 3 | 14.4 | 20.8 | 1335.7 | 2729.2 | 11.3 |
|  | 2 | 142.2 | 498.3 | 2292.4 | 13476.7 | 16.9 |  |  | 4 | 145.3 | 195.2 | 1142.8 | 2313.4 | 14.1 |
|  | 3 | 140.5 | 622.1 | 2522.3 | 18777.3 | 21.3 |  |  | 5 | 209.0 | 282.8 | 1840.7 | 3023.6 | 14.3 |
|  | 4 | 222.6 | 1085.6 | 2537.6 | 20031.5 | 24.2 |  | 51 | 1 | 231.7 | 357.6 | 2822.4 | 5368.9 | 10.5 |
|  | 5 | 302.3 | 1916.6 | 3490.5 | 21924.8 | 30.3 |  |  | 2 | 441.1 | 630.5 | 2454.0 | 4946.7 | 11.5 |
| 101 | 1 | 463.1 | 3773.2 | 5407.8 | 33954.8 | 21.1 |  |  | 3 | 453.1 | 663.2 | 2902.1 | 5440.4 | 11.9 |
|  | 2 | 347.5 | 2288.8 | 4789.9 | 36727.3 | 22.2 |  |  | 4 | 227.9 | 358.9 | 2736.7 | 4679.5 | 11.1 |
|  | 3 | 541.4 | 4926.1 | 6004.2 | 40479.8 | 25.7 |  |  | 5 | 27.2 | 58.2 | 3186.7 | 4323.9 | 9.1 |
|  | 4 | 398.0 | 2889.1 | 5001.0 | 37140.0 | 21.8 |  | 101 | 1 | 476.6 | 781.5 | 5086.1 | 9504.9 | 9.3 |
|  | 5 | 461.4 | 3398.0 | 6625.2 | 40491.0 | 27.3 |  |  | 2 | 662.4 | 1006.1 | 5507.3 | 9885.5 | 10.1 |
| 201 | 1 | 950.0 | 11324.7 | 9809.7 | 68533.0 | 21.0 |  |  | 3 | 54.8 | 148.5 | 5626.3 | 8105.5 | 7.8 |
|  | 2 | 979.3 | 12219.7 | 10539.0 | 71759.8 | 22.9 |  |  | 4 | 858.7 | 1320.1 | 6437.0 | 10034.1 | 10.1 |
|  | 3 | 997.9 | 12035.4 | 11843.0 | 76078.0 | 24.8 |  |  | 5 | 737.0 | 1177.6 | 5190.0 | 9748.5 | 9.9 |
|  | 4 | 910.5 | 11935.3 | 11438.4 | 76861.6 | 24.2 |  | 201 | 1 | 1000.5 | 1900.2 | 10170.1 | 20245.3 | 9.0 |
|  | 5 | 693.9 | 6598.2 | 9920.7 | 57901.8 | 18.0 |  |  | 2 | 159.3 | 590.0 | 11884.4 | 17890.7 | 8.1 |
| 501 | 1 | 2750.0 | 71994.4 | 24948.2 | 176894.0 | 22.4 |  |  | 3 | 881.2 | 1728.8 | 9777.2 | 20692.0 | 8.7 |
|  | 2 | 2465.4 | 61735.4 | 26412.1 | 170807.0 | 21.5 |  |  | 4 | 802.8 | 1561.9 | 10312.3 | 19110.1 | 8.3 |
|  | 3 | 2350.1 | 72256.9 | 24382.1 | 173560.0 | 22.3 |  |  | 5 | 1946.2 | 3110.6 | 10868.0 | 22597.3 | 9.0 |
|  | 4 | 2821.9 | 71924.2 | 22053.0 | 167324.0 | 21.0 |  | 501 | 1 | 1827.2 | 5095.0 | 26988.8 | 46531.6 | 7.8 |
|  | 5 | 2742.2 | 72347.3 | 23655.0 | 164094.0 | 21.0 |  |  | 2 | 3432.9 | 7138.1 | 27202.1 | 46359.8 | 8.3 |
|  |  |  |  |  |  |  |  |  | 3 | 2094.9 | 5431.2 | 29882.9 | 47885.6 | 7.7 |
|  |  |  |  |  |  |  |  |  | 4 | 2719.3 | 6828.1 | 30291.2 | 44805.7 | 8.2 |
|  |  |  |  |  |  |  |  |  | 5 | 2738.2 | 6256.0 | 28583.7 | 46866.3 | 7.8 |

*𝑛𝑇*

MDA∗ A∗ LSET LCOR #p

1. Finally, time windows [*𝑒𝑠, 𝑙𝑠*] are defined for all customers *𝑠* ∈ *𝑇* ⧵

{*𝑠*0}*,* centered on the visit time in the greedy solution and with a

width equal to *𝑙*0 (truncated to make sure that *𝑒* ≥ 0 and *𝑙* ≤ *𝑙* ).

*𝑠* 0

5

*𝑠*

“|*𝑇* +(*𝑣*0)|” reports the average number of reachable destinations; Col- Results are presented in [Tables 15](#_bookmark28) and [16](#_bookmark29). In these tables, Column

umn “# removed” indicates the average number of nodes that could be removed. Column “CPU %” indicates the average impact on computing time (in %), including preprocessing. Column “#p” indicates the average impact on the number of eﬃcient paths found between two terminals (in %). Note that this gap is not null because it compares the number of paths with or without time windows.

ber of considered destinations |*𝑇* +(*𝑣*0)| is significantly reduced: approxi- From these tables, we notice that, with our enhancements, the num-

**Table 15**

Impact of enhancements on network AIX-2 - Ran- dom.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *𝑛𝑇* | |*𝑇* +(*𝑣*0 )| | # removed | CPU % | #p % |
| 26 | 13 | 2881 | -50.3 | -64.0 |
| 51 | 26 | 3766 | -41.3 | -61.1 |
| 101 | 51 | 3373 | -37.6 | -63.1 |
| 201 | 101 | 3311 | -38.4 | -63.6 |
| 501 | 255 | 2902 | -26.3 | -61.8 |

**Table 16**

Impact of enhancements on network DC.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *𝑛𝑇* | |*𝑇* +(*𝑣*0 )| | # removed | CPU % | #p % |
| 26 | 14 | 1245 | -32.4 | -52.8 |
| 51 | 27 | 1778 | -41.8 | -54.2 |
| 101 | 51 | 1821 | -37.7 | -56.6 |
| 201 | 95 | 1984 | -37.4 | -60.8 |
| 501 | 239 | 2000 | -32.2 | -60.6 |

mately divided by 2 on average. We also see that the number of removed nodes is relatively important. Regarding computing times, it clearly ap- pears that the algorithm is faster when the enhancements are used. Re- garding the number of paths, one can observe that the size of complete sets of eﬃcient paths is divided by more than 2.

# Conclusion

In this paper, we introduced and investigated the Steiner bi-objective shortest path problem. The particularity of this problem is to seek for complete set of eﬃcient paths linking a subset of nodes in the net- work, the so-called terminals. We developed an exact solution approach

based on a labeling algorithm combined with a modified *𝐴*∗ algorithm.

The proposed approach is based on a goal-directed search strategy that

guides labels simultaneously towards all terminals and allows stopping the search quickly once all eﬃcient paths have been found. The moti- vation for this new problem stems from the preprocessing of travel in- formation for vehicle routing problems and we proposed and evaluated several enhancements in the case of the VRPTW.

Computational experiments were carried out on a large panel of in- stances based on real road networks. Results show that the proposed al-

gorithm performs very well for all tested instances. The MDA∗ algorithm

largely outperforms state-of-the-art algorithms. Additional savings are

obtained when taking into account time windows.

Experiments also give insights on the solution of vehicle routing problems with multigraphs. They show that multigraphs of acceptable size should be obtained and that these graphs could be computed in a reasonable amount of time for road-network graphs with dozens of thousand of nodes and hundreds of customers.

Would these computing times be considered too high in practice, several possible adaptations of our algorithms could be possible. For example, computations might be simplified a lot by only keeping the min-time and min-cost paths. Also, some heuristic rules might be intro- duced in the dynamic programming algorithm to limit the number of labels. Mono-objective shortest path problems with different weighted sums for travel time and cost could also be solved. Of course, with all these approaches, the theoretical guarantee of optimality would be lost for the subsequent vehicle routing problem.

# Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

# Acknowledgements

The first author was supported by the Labex IMobS3, by the Euro- pean Fund for Regional Development (FEDER Auvergne region) and by the Auvergne Region. The authors thank anonymous referees for their careful reading, thorough reviews and valuable comments that helped us improving the quality of the paper.

# References

[Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wag-](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0001) [ner, D., Werneck, R.F., 2016. Route planning in transportation networks. In: Algo- rithm Engineering. Springer, pp. 19–80.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0001)

[Ben](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0002) [Ticha, H., Absi, N., Feillet, D., Quilliot, A., 2017. Empirical analysis for the VRPTW with a multigraph representation for the road network. Comput. Oper. Res. 88, 103–116.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0002)

[Ben](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0003) [Ticha, H., Absi, N., Feillet, D., Quilliot, A., 2018. Vehicle routing problems with road-](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0003)

[-network information: State of the art. Networks 72 (3), 393–406.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0003)

[Brumbaugh-Smith, J., Shier, D., 1989. An empirical investigation of some bicriterion short- est path algorithms. Eur. J. Oper. Res. 43 (2), 216–224.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0004)

[Clímaco,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0005) [J.C., Pascoal, M., 2012. Multicriteria path and tree problems: Discussion on exact algorithms and applications. Int. Trans. Oper. Res. 19 (1-2), 63–98.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0005)

[Climaco,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0006) [J.C.N., Martins, E.Q.V., 1982. A bicriterion shortest path algorithm. Eur. J. Oper.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0006)

[Res. 11 (4), 399–404.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0006)

[Cornuéjols,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0007) [G., Fonlupt, J., Naddef, D., 1985. The traveling salesman problem on a graph and some related integer polyhedra. Math. Program. 33 (1), 1–27.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0007)

[Dijkstra,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0008) [E.W., 1959. A note on two problems in connexion with graphs. Numerische Math- ematik 1 (1), 269–271.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0008)

[Ehrgott,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0009) [M., Gandibleux, X., 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum 22 (4), 425–460.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0009)

[Feillet,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0010) [D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the ele- mentary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44 (3), 216–229.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0010)

[Floyd, R.W., 1962. Algorithm 97: Shortest path. Commun. ACM 5 (6), 345](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0011).

[Garaix,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0012) [T., Artigues, C., Feillet, D., Josselin, D., 2010. Vehicle routing problems with al- ternative paths: An application to on-demand transportation. Eur. J. Oper. Res. 204 (1), 62–75.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0012)

[Ghoseiri,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0013) [K., Nadjari, B., 2010. An ant colony optimization algorithm for the bi-objective shortest path problem. Appl. Soft Comput. 10 (4), 1237–1246.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0013)

[Guerriero,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0014) [F., Musmanno, R., 2001. Label correcting methods to solve multicriteria short- est path problems. J. Optim. Theory Appl. 111 (3), 589–613.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0014)

[Hansen,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0015) [P., 1980. Bicriterion path problems. In: Multiple Criteria Decision Making Theory and Application. Springer, pp. 109–127.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0015)

[Hart,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0016) [P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0016)

[Huang,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0017) [Y., Zhao, L., Van Woensel, T., Gross, J.-P., 2017. Time-dependent vehicle routing problem with path flexibility. Transportation Research Part B: Methodological 95, 169–195.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0017)

[Huarng,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0018) [F., Pulat, P., Shih, L., 1996. A computational comparison of some bicriterion shortest path algorithms. J. Chin. Inst. Ind. Eng. 13 (2), 121–125.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0018)

[Irnich,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0019) [S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Column Generation. Springer, pp. 33–65.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0019)

[Lai,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0020) [D.S., Demirag, O.C., Leung, J.M., 2016. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transp. Res. Part E: Logist. Transp. Rev. 86, 32–52.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0020)

[Letchford,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0021) [A.N., Nasiri, S.D., Oukil, A., 2014. Pricing routines for vehicle routing with time windows on road networks. Comput. Oper. Res. 51, 331–337.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0021)

[Letchford,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0022) [A.N., Nasiri, S.D., Theis, D.O., 2013. Compact formulations of the Steiner trav- eling salesman problem and related problems. Eur. J. Oper. Res. 228 (1), 83–92.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0022)

[Mandow,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0023) [L., Pérez de la Cruz, J.L., 2009. A memory-eﬃcient search strategy for mul- tiobjective shortest path problems. In: Annual Conference on Artificial Intelligence. Springer, pp. 25–32.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0023)

[Mandow,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0024) [L., Pérez de la Cruz, J.L., 2010. Multiobjective A∗ search with consistent heuris-](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0024)

[tics. J. ACM (JACM) 57 (5), 27.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0024)

[Martins,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0025) [E.Q.V., 1984. An algorithm for ranking paths that may contain cycles. Eur. J. Oper. Res. 18 (1), 123–130.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0025)

[Martins,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0026) [E.Q.V., 1984. On a multicriteria shortest path problem. Eur. J. Oper. Res. 16 (2), 236–245.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0026)

Martins, E. Q. V., Santos, J., 1999. The labeling algorithm for the multiobjective short- est path problem. Departamento de Matematica, Universidade de Coimbra, Portugal, Tech. Rep. TR-99/005.

[Mote,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0028) [J., Murthy, I., Olson, D.L., 1991. A parametric approach to solving bicriterion short- est path problems. Eur. J. Oper. Res. 53 (1), 81–92.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0028)

[Müller-Hannemann, M., Weihe, K., 2006. On the cardinality of the Pareto set in bicriteria shortest path problems. Ann. Oper. Res. 147 (1), 269–286.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0029)

[Paixão,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0030) [J.M., Santos, J.L., 2013. Labeling methods for the general case of the multi-ob- jective shortest path problem–a computational study. In: Computational Intelligence and Decision Making. Springer, pp. 489–502.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0030)

[Pangilinan,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0031) [J.M.A., Janssens, G.K., 2007. Evolutionary algorithms for the multiobjective shortest path problem. World Acad. Sci. Eng. Technol. Int. J. Math. Comput. Phys. Electr. Comput. Eng. 1 (1), 7–12.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0031)

[Raith,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0032) [A., Ehrgott, M., 2009. A comparison of solution strategies for biobjective shortest path problems. Compute. Oper. Res. 36 (4), 1299–1331.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0032)

Schultes, D., 2005. Tiger road networks for 9th DIMACS implementation challenge– shortest path. <http://users.diag.uniroma1.it/challenge9/data/tiger>.

[Serafini,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0034) [P., 1987. Some considerations about computational complexity for multi ob- jective combinatorial problems. In: Recent Advances and Historical Development of Vector Optimization. Springer, pp. 222–232.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0034)

[Skriver,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0035) [A.J., 2000. A classification of bicriterion shortest path (BSP) algorithms. Asia Pac.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0035)

[J. Oper. Res. 17 (2), 199–212.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0035)

[Stewart, B.S.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0036), [White III, C.C., 1991. Multiobjective A∗. J. ACM (JACM) 38 (4), 775–814](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0036).

[Tung,](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0037) [C.T., Chew, K.L., 1992. A multicriteria Pareto-optimal path algorithm. Eur. J. Oper.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0037)

[Res. 62 (2), 203–209.](http://refhub.elsevier.com/S2192-4406(21)00001-0/sbref0037)