Exact Methods for Mono-Objective and Bi-Objective Multi-Vehicle Covering Tour Problems

Estèle Glize, Roberto Roberti, Nicolas Jozefowiez, Sandra Ulrich Ngueveu

Exact Methods for Mono-Objective and Bi-Objective Multi-Vehicle Covering Tour Problems

Estèle Glize[a,b], Roberto Roberti[c], Nicolas Jozefowiez[d], Sandra Ulrich Ngueveu[a,b]

[a]*CNRS, LAAS, 7 avenue du colonel Roche, F-31077 Toulouse, France*
[b]*Universite de Toulouse, INSA, INP, LAAS, F-31077 Toulouse, France*
glize@laas.fr & ngueveu@laas.fr
[c]*Department of Supply Chain Analytics, Vrije Universiteit Amsterdam, Amsterdam, 1081HV, the Netherlands*
r.roberti@vu.nl
[d]*LCOMS, Université de Lorraine, Metz, France*
nicolas.jozefowiez@univ-lorraine.fr

Abstract

The *Multi-vehicle Covering Tour Problem* and the *Bi-Objective Multi-vehicle Covering Tour Problem* have been studied for more than thirty years. Both problems have several practical applications in industry. In this paper, we propose an effective exact method for the *Multi-vehicle Covering Tour Problem* based on column generation techniques. The effectiveness of the exact method is owed to tailored dominance rules and completion bounds. To validate our approach, we conducted extensive computational experiments on instances from literature. The comparison with state-of-the-art methods shows the effectiveness of the proposed method. In particular, seven open instances are closed to optimality for the first time, and the best lower bounds of the six open instances are improved. The exact method for the *Multi-vehicle Covering Tour Problem* is also embedded in a ϵ-constraint exact method to solve its bi-objective counterpart. Computational results show that the lower bound set provided by this bi-objective exact method is stronger than those provided by the state-of-the-art method from the literature.

Keywords: Combinatorial Optimization, Multi-Vehicle Covering Tour Problem, Bi-objective optimization, Vehicle Routing Problems, Column Generation

1. Introduction

The *Covering Tour Problem* (CTP) was introduced by Current and Schilling (1989) as a generalization of the *Traveling Salesman Problem* where not all cities of the network have to be visited. However, they must be within some predetermined covering distance from the nearest visited city. Current and Schilling (1994) view the CTP as the design of a bi-level transportation networks. A main vehicle performs a minimum length tour to serve some subsidiary points, and any point that does not belong to

the tour can reach the nearest visited point in decent time. In the *Multi-vehicle Covering Tour Problem* (MCTP), several vehicles are available to perform the tours.

Covering tour problems have interesting real-life applications, such as, humanitarian logistics, regional development, urban patrolling, and health-care management. In humanitarian logistics, Naji-Azimi et al. (2012) and Nolz et al. (2010) perform survival goods distribution throughout disaster area; the relief teams can only visit a few satellite distribution centers, and the population directly goes to these centers if they are close enough from their homes. In regional development, Current and Schilling (1994) and Labbé and Laporte (1986) determine locations for overnight mail service accessible for the complete population and for post boxes. Oliveira et al. (2015) plan routes for urban patrolling that pass through some important points,such as schools and hospitals and near some other convenient points, such as public parks and bank agencies. Hodgson et al. (1998) and Hachicha et al. (2000) both manage the health care in rural areas where a medical team is supposed to visit a small subset of villages so that each inhabitant is able to reach its nearest village in an acceptable amount of time.

1.1. Problem Description

The MCTP aims at finding the minimum-cost routes over a network that fulfill the requirements of different nodes. Some nodes must be visited by the vehicles whereas some other nodes have to be covered. A node is covered if it is situated within a predefined covering distance from its nearest visited node.

Let $G = (V, E)$ be an undirected graph. The vertex set V is defined as $V = \{0\} \cup M \cup O \cup C$, where 0 is the depot, M and O are respectively the sets of *mandatory* and *optional service points* (i.e., nodes to visit), and C is the set of *customers* (i.e., nodes to cover). The edge set E is defined as $E = \{\{i, j\} | i, j \in V, i < j\}$. A distance d_{ij} is associated with each edge $\{i, j\} \in E$. At the depot, a set K of $|K|$ homogeneous vehicles is located. The vehicles have to perform a set of routes that visits the mandatory service points, and can visit the optional service points. A visit to an optional service point $i \in O$ allows to cover all customers $j \in C$ such that $d_{ij} \leq \eta$, where η is the maximum coverage distance. If a route r visits a service point $i \in O$ that covers customer $j \in C$, we say that route r covers customer j. The customers must be covered by at least one route. Each route cannot visit more than $\overline{n}$ service points, and the sum of the distances of the edges traversed by a route cannot exceed the maximum distance $\overline{d}$. The goal of the MCTP is to design a set of routes such that the total distance of the routes is minimized, the mandatory service points are visited, and the customers are covered. The MCTP is an NP-hard problem as the MCTP reduces to the *Distance*

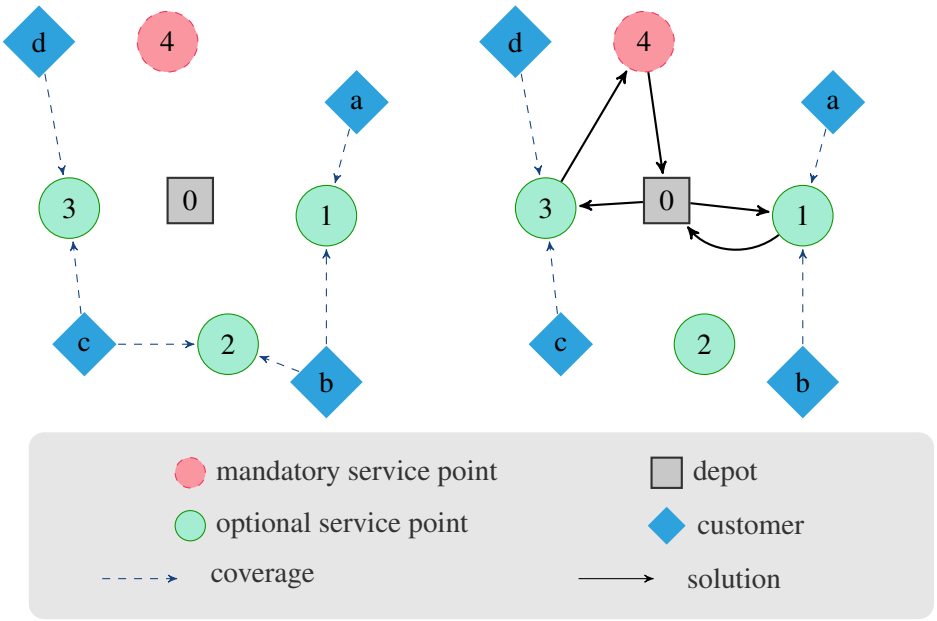

Figure 1: Example of an MCTP with one mandatory service point, three optional service points, and four customers: the corresponding graph on the left panel, and a feasible solution on the right panel

Constrained Vehicle Routing Problem (Toth and Tramontani (2008)) if $V = M$ and cardinality constraints are relaxed or to the *Vehicle Routing Problem with unit demands* (Godinho et al. (2008)) if $V = M$ and maximum distance constraints are relaxed.

Figure 1 provides an example of an MCTP and a corresponding feasible solution. The graph of the MCTP is presented on the left panel. The depot is the node 0, the service points are $\{1, 2, 3, 4\}$ and the customers are the set $\{a, b, c, d\}$. A dashed blue arrow links an optional service point $i \in O$ and a customer $j \in C$ if j is covered by i (if visited). On the right panel, the black arrows represent a solution composed of two routes $0 \to 1 \to 0$ and $0 \to 3 \to 4 \to 0$. This solution is feasible as the mandatory service point 4 is visited, and all customers are covered by the routes.

The *Bi-Objective Multi-vehicle Covering Tour Problem* (BOMCTP) also studied in this paper aims to minimize the total distance of the routes as first objective and considers the maximum coverage distance η as a second criterion to minimize. All other input data remain the same as in the MCTP. These two objectives are conflicting since the larger the value of η, the fewer the number of nodes to visit, and therefore the smaller the distance to travel.

1.2. Literature

For the sake of clarity, the literature about covering tour problems is divided into four parts. We discuss the literature on the CTP, the MCTP, the BOCTP, and the BOMCTP in Section 1.2.1, Section 1.2.2, Section 1.2.3, and Section 1.2.4, respectively.

1.2.1. Covering Tour Problem

Current and Schilling (1989) formulate the *Covering Salesman Problem* (CSP) with a two-index formulation. The CSP is similar to the CTP except than the customers are the optional service points that can be visited and there is no mandatory service points. They devise a heuristic that solves a *Set Covering Problem* (SCP) on the original graph and a *Traveling Salesman Problem* on the resulting graph of each optimal solution of the SCP. They test their heuristic on an instance with 21 nodes and 39 edges.

Gendreau et al. (1997) propose valid inequalities as well as a branch-and-cut method to solve a two-index formulation for the CTP. They also introduce a heuristic to find an upper bound at the root node that generally provides solution within 3% from optimality. They test their exact method on randomly generated instances with up to 600 nodes and up to 100 service points and solve them in less than two hours.

Baldacci et al. (2005) formulate the CTP as a two-commodity network flow problem. They compare their new formulation with the one proposed by Gendreau et al. (1997) and show that the corresponding lower bound is tighter. They develop a scatter-search metaheuristic and compare a classic scatter search method with a version that generates violated generalized subtour elimination constraints. They create random instances in the same way of Gendreau et al. (1997) and with the same size and solved them in 20 minutes at most.

1.2.2. Multi-vehicle Covering Tour Problem

The MCTP was introduced by Hachicha et al. (2000) who formulate the problem with a three-index formulation. They apply three different heuristics for the MCTP: the savings heuristic of Clarke and Wright (1964), the sweep algorithm of Gillett and Miller (1974), and the route-first/cluster-second heuristic of Beasley (1983). Each heuristic uses the one developed by Gendreau et al. (1997) for the CTP. The algorithms are tested on randomly generated instances with up to 200 service points and 400 customers. These instances are solved in less than half-an-hour by the modified sweep algorithm and in less than 4 minutes by the two others heuristics. The route-first/cluster-second heuristic also solves real-life in-

stances in few seconds. The computational experiments show that the route-first/cluster-second heuristic is the best compromise between time and quality of the solution.

Lopes et al. (2013) present preliminary results of a branch-and-price algorithm for the MCTP based on a new set-partitioning formulation. The solution of the sub-problem is accelerated with a GRASP metaheuristic. Computational experiments are done on randomly generated instances created as in Hachicha et al. (2000). They set the maximum distance $\overline{d}$ equal to 200 and the maximum number of service points $\overline{n}$ to 4. Only 5 instances out of 15 are solved in less than 4 hours on instances with up to 400 nodes and with up to 200 service points.

Jozefowiez (2014) presents a set partitioning formulation for the MCTP with fewer constraints than Lopes et al. (2013). He develops a branch-and-price algorithm in which the sub-problems are formulated as a ring star problem and solved by a branch-and-cut algorithm. The instances are randomly generated with up to 60 service points and with up to 150 customers. He chooses to set the maximum distance $\overline{d}$ to infinity and to vary the maximum number of service points $\overline{n}$ between 5 and 8. The bounds of that formulation are tight, but they take too long to be evaluated.

Ha et al. (2013) propose a new two-commodity flow formulation for the MCTP based on the formulation of Baldacci et al. (2005) for the CTP and some valid inequalities applied for the MCTP. They solve their formulation with a branch-and-cut method. They also devise a two-phase metaheuristic which first randomly creates subsets of service points that can cover all customers and then solves a *Vehicle Routing Problem* with unit demand on each subset. This metaheuristic provides the initial upper bounds for their branch-and-cut algorithm. They tested their exact method with the same instances of Jozefowiez (2014). Their results outperform previous works on exact methods for the MCTP, so their exact method is currently the state-of-the-art exact method from the literature. Therefore, we will compare their results with the results achieved by our exact method in Section 4.

Murakami (2014) introduces a column generation-based heuristic for the MCTP. The heuristic outperforms the heuristics of Hachicha et al. (2000) in terms of cost but not in terms of time. He uses the same methods applied for the *Generalized Multi-Vehicle Covering Tour Problem* in Murakami (2018) where the service points have a demand that can exceed one. The results show the efficiency of the heuristic based on column generation when compared to Hachicha et al. (2000) on the instances of Jozefowiez (2014). The algorithm applied to new randomly generated instances with up to 400 service points shows good results in reasonable time.

Kammoun et al. (2017) apply a variable neighborhood search heuristic to the MCTP. They compare their metaheuristic to the metaheuristic of Ha et al. (2013) on the same instances. Their method obtains better or equal results than the one of Ha et al. (2013) and faster. The best-known upper bounds of 4 of the 96 instances are improved. They generate new instances derived from the TSPLIB instances with up to 700 nodes with 362 service points and show that the metaheuristic can provide good solutions in a few seconds of computing time..

Recently, Pham et al. (2017) introduced the *Multi-vehicle Multi-Covering Tour Problem* (MMCTP) where customers have to be covered a certain number of times. They adapt the formulation and the metaheuristic of Ha et al. (2013) to the MMCTP. Randomly generated test instances are created following the process of Jozefowiez (2014) and, in addition, choosing the number of times customers needs to be visited. The instances involve up to 200 nodes with 50 service points.

Flores-Garza et al. (2017) also introduce a variation of the MCTP called the *Multi-vehicle Cumulative Covering Tour Problem* that minimizes the sum of arrival times at each service point. They use a three-index formulation to solve their problem as well as a GRASP method. They test the efficiency of their heuristic on the same instances of Jozefowiez (2014).

1.2.3. Bi-Objective Covering Tour Problem

Current and Schilling (1994) present two problems called the *Median Tour Problem* and the *Maximal Covering Tour Problem* that are bi-objective versions of the CSP presented in Current and Schilling (1989). The two problems aim to minimize the total tour length. The *Median Tour Problem* also minimizes the total travel distance of unvisited nodes to their nearest visited node weighted by their demands. On the other hand, the *Maximal Covering Tour Problem* minimizes the demand of nodes not covered by the tour. They formulate them with a two-index formulation and devised heuristics to solve them. A first solution that minimizes only one objective is selected as initial solution. The Pareto front is completed by moving along the front to adjacent solutions in the sense that optimizes the other objectives. They use a real-life instance of 681 nodes to test their algorithm. An average of 26 points are found in the Pareto front depending on the initial solution.

Jozefowiez et al. (2007) introduce another bi-objective version of the CTP where the first objective is not changed and the other one aims to minimize the maximum coverage distance η. Jozefowiez et al. (2007) design a two-phase metaheuristic that uses the solution of a multi-objective evolutionary algorithm as input for the branch-and-cut of Gendreau et al. (1997). They make computational experiments

on randomly generated data with up to 120 service points and 360 customers and a real-life instance of Hodgson et al. (1998). They compare their metaheuristic with an exact method: the branch-and-cut algorithm of Gendreau et al. (1997) embedded in the ϵ-constraint method. The metaheuristic gives good quality solutions and runs faster on larger instances.

Nolz et al. (2010) propose a heuristic based on genetic algorithms and variable neighborhood search to solve a BOCTP that minimizes the distance between uncovered customer and the nearest visited service point and that minimizes either the tour length or the latest arrival time at a service point. They use real-life instances for the experiments with up to 41 service points.

1.2.4. Bi-Objective Multi-Vehicle Covering Tour Problem

Tricoire et al. (2012) introduce a bi-objective stochastic multi-vehicle covering tour problem that minimizes both the total tour length plus the cost of the distribution centers opening and the expected uncovered demand. The demand of customers are uncertain and are approximated by an empirical sample distribution. Each service point has a maximal capacity of affected customers. They embed a branch-and-cut technique in an ϵ-constraint method to solve their BOMCTP. A heuristic based on the ϵ-constraint algorithm is also devised that allows a fixed gap tolerance for each optimal solution of the Pareto front. Real-life data are used to test their method with up to 30 nodes and a time limit of three days. The Pareto front can vary between 6 to 112 optimal solutions. Their heuristic seems to give a good approximation of the Pareto front within reasonable times.

Oliveira et al. (2015) plan urban patrolling routes in which the total distance of the selected routes is minimized whereas the second objective aims to maximize the fairness between the routes in terms of number of customers visited. They transform the second objective into a constraint and apply different heuristics to the instances of Ha et al. (2013) and real-life instances.

Artigues et al. (2018) construct the lower bound set of a bi-objective version of the MCTP where the first objective is to minimize the total routes length and the other one minimizes the maximum coverage distance η. They use column generation techniques embedded in an ϵ-constraint method. To the best of our knowledge, no exact method on the bi-objective MCTP as formulated in Artigues et al. (2018) has been proposed in the literature.

Table 1 aggregates the different papers on the BOCTP and the BOMCTP found in literature. The second column $|K|$ indicates the number of available vehicles at the depot. The remaining columns represent the different objectives used the literature, namely:

- *Total cost*: minimization of the total cost of the routes;
- *Customers traveling distance*: minimization of the total travel distance between unvisited nodes and their nearest visited node;
- *Uncovered demand*: minimization of the demands not covered by the routes;
- *Maximum coverage distance*: minimization of the maximum coverage distance;
- *Latest arrival*: minimization of the latest arrival time at a service point;
- *Fairness*: maximization of the fairness between the routes.

Our study on the BOMCTP will focus on the two objective functions investigated by Artigues et al. (2018).

Table 1: Table on the literature of the BOCTP

Authors	$\|K\|$	Total cost	Customers traveling distance	Uncovered demand	Maximum coverage distance	Latest arrival	Fairness
Current and Schilling (1994)	1	x	x				
	1	x		x			
Jozefowiez et al. (2007)	1	x			x		
Nolz et al. (2010)	1	x	x				
	1		x			x	
Tricoire et al. (2012)	m	x		x			
Oliveira et al. (2015)	m	x					x
Artigues et al. (2018)	m	x			x		

1.3. Overview of the paper

This paper aims to solve both the MCTP as studied by Hachicha et al. (2000), Lopes et al. (2013), Jozefowiez (2014), Ha et al. (2013), Murakami (2014) and Kammoun et al. (2017) and the BOMCTP introduced by Artigues et al. (2018) that aim to minimize both the total distance of the routes and the maximum coverage distance η. A new state-of-the-art exact algorithm is proposed, and it strictly outperforms existing methods from the literature. It is also applied successfully to solve the bi-objective variant of the problem.

Section 2 presents the mathematical formulation, the exact method used to solve MCTP and the main ingredients of the exact method. Section 3 explains how to embed our method in an ϵ-constraint method to solve the BOMCTP. Section 4 shows the computational results of the proposed methods on the MCTP and the BOMCTP. Finally, Section 5 summarizes some conclusions and future research directions.

2. An Exact Method for the MCTP

In this section, we present a mathematical formulation of the MCTP and describe the exact method we propose to solve the MCTP to optimality.

2.1. Mathematical Model

The following formulation of the MCTP has been introduced in Lopes et al. (2013). Let R be the set of all feasible routes. Each vehicle performs a route $r \in R$ that starts at the depot, visits some service points denoted by S_r, and returns to the depot. The cardinality of r is denoted by q_r. A route is said to be feasible if it does not visit more than $\overline{n}$ service points, i.e., $q_r \leq \overline{n}$, and the sum of the distance of the traversed edges, denoted by d_r, does not exceed the maximum distance $\overline{d}$. For each route $r \in R$ and each service point $i \in M \cup O$, let a_{ir} be a binary coefficient equal to 1 if route r visits service point i. Moreover, for each route $r \in R$, let $C_r \subseteq C$ be the subset of customers covered by r (i.e., by the visited service points S_r). For each customer $i \in C$, let b_{ir} be a binary coefficient equal to 1 if $i \in C_r$ and 0 otherwise. Let us introduce a binary variable $x_r \in \{0, 1\}$, $r \in R$, which indicates if the route is selected ($x_r = 1$) or not ($x_r = 0$). The MCTP can be formulated as follows [Lopes et al. 2013]:

$$\min z = \sum_{r \in R} d_r x_r \tag{1}$$

$$s.t. \sum_{r \in R} a_{ir} x_r = 1 \qquad i \in M \tag{2}$$

$$\sum_{r \in R} a_{ir} x_r \leq 1 \qquad i \in O \tag{3}$$

$$\sum_{r \in R} b_{ir} x_r \geq 1 \qquad i \in C \tag{4}$$

$$\sum_{r \in R} x_r \leq |K| \tag{5}$$

$$x_r \in \{0, 1\} \qquad r \in R \tag{6}$$

The objective function (1) minimizes the total distance of the selected routes. Constraints (2) ensure that all mandatory service points are visited whereas constraints (3) allow to visit optional service points if necessary. Due to constraints (4), every customer has to be covered at least once. Constraint (5) guarantees that at most $|K|$ routes are selected. Finally, constraints (6) define the range of the decision variables.

Let us notice that constraints (3) are redundant if the distance matrix $\boldsymbol{d}$ satisfies the triangular inequality and the graph G is complete. Otherwise, constraints (3) are necessary unless a solution can visit

a service point without serving it. Some papers in the literature keep such constraints (Hachicha et al. (2000), Lopes et al. (2013), Ha et al. (2013)) whereas some others do not (Jozefowiez (2014), Artigues et al. (2018)). We keep the constraints as we do not wish to make hypothesis on the instances and design a generic algorithm.

The exact method used to solve the MCTP is based on formulation (1)-(6) and on the exact method proposed in Baldacci et al. (2008) for the *Capacitated Vehicle Routing Problem*. It consists of four main steps:

1. By using column generation, compute a lower bound LB corresponding to the optimal solution of the linear relaxation of formulation (1)-(6) and its associated dual solution.
2. Compute an upper bound UB to the MCTP.
3. Generate the set $\overline{R}$ of all feasible routes with a reduced cost, with respect to the dual solution computed at Step 1, less than or equal to the gap, $UB - LB$.
4. Solve formulation (1)-(6) with the set $\overline{R}$ to optimality with a general-purpose MILP solver.

The four steps of the exact method are described in the next sections, 2.1-2.4. In the following, let $\lambda_i \in \mathbb{R}$ ($i \in M$), $\gamma_j \leq 0$ ($j \in O$), $\sigma_k \geq 0$ ($k \in C$) and, $\gamma_0 \leq 0$ be the duals associated to constraints (2), (3), (4) and (5), respectively. Furthermore, let us denote by *Master Problem* (MP) the linear relaxation of formulation (1)-(6) and by *Restricted Master Problem* (RMP) the Master Problem with a restricted set of columns.

2.2. Step 1: Computing Lower Bound LB

The lower bound LB is computed by using column generation. The set of routes $\tilde{R}$ is initialized with all feasible routes of cardinality one and two plus an artificial route that visits all service points and has a large enough cost $\tilde{d}$.

The pricing sub-problem for the MCTP corresponds to an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC) which is known to be strongly NP-hard (Dror (1994)) and can therefore be time-consuming to solve. To accelerate the solution of the ESPPRC, we apply the ng-route relaxation (introduced by Baldacci et al. (2011)) and propose tailored dominance rules and completion bounds as well as a stabilization technique and a heuristic algorithm to efficiently price out columns.

2.2.1. Ng-routes relaxation

The ng-routes relaxation, introduced by Baldacci et al. (2011), is an efficient relaxation of the ESPPRC which obtains near-elementary routes in limited computing times. This relaxation usually allows to construct routes containing long cycles in terms of distances on the edges. By construction, short cycles are discarded. The new pricing sub-problem solved when pricing out ng-routes with negative reduced cost is the *ng-Shortest Path Problem with Resource Constraints* (ng-SPPRC).

The ng-routes relaxation can be summarized as follows. For each service point i, a set N_i of cardinality Δ is pre-defined. Such a set N_i contains i plus the $\Delta - 1$ service points that are closest to i.

For each path $p = (0, i_1, i_2, .., i_{n_p})$ that starts at the depot, visits a set of service points $i_1, i_2, .., i_{n_p}$, and ends at service point i_{n_p}, an ng-set $NG_p \subseteq N_{i_{n_p}}$ is computed as follows:

$$NG_p = \{i_k, k = 1, .., n_p - 1 | i_k \in \cap_{j=k+1}^{n_p} N_{i_j}\} \cup \{i_{n_p}\}$$

An ng-set contains the forbidden service points, i.e., the points that have already been visited by the path and that are considered too close to return to. A feasible ng-path is a non-necessarily elementary path $p = (0, i_1, i_2, .., i_{n_p})$ that starts at the depot, satisfies the maximum cardinality constraint, the maximum distance constraint and the ng-elementarity, and ends at the service point i_{n_p}. Any ng-path where $i_{n_p} = 0$ is a feasible ng-route. The set R of feasible routes in formulation (1)-(6) is now replaced with the set R' of feasible ng-routes.

2.2.2. Labeling algorithm

Solving the ng-SPPRC consists of dynamically generating routes with a negative reduced cost with respect to the dual solution $(\lambda, \gamma, \sigma)$ of RMP via a labeling algorithm. A label L is a feasible ng-path that starts at the depot, visits a set of service points S_L, covers the customers C_L, ends at the service point v_L with an ng-set NG_L, has cardinality q_L, traveling distance τ_L and reduced cost c_L. Let us define the reduced cost of edge $\{i, j\}$ as:

$$d'_{ij} = d_{ij} - (e_i + e_j)/2 \qquad \forall \{i, j\} \in E, i \in O \cup M \cup \{0\} \tag{7}$$

with

$$e_i = \begin{cases} \lambda_i & i \in M \\ \gamma_i & i \in O \\ \gamma_0 & i = 0 \end{cases}$$

The label extension of L from v_L to a service point $w \in \{O \cup M\} \backslash NG_L$ gives the new label L' according to propagation rules (8)-(13):

$$v_{L'} = w \tag{8}$$

$$q_{L'} = q_L + 1 \tag{9}$$

$$\tau'_L = \tau_L + d_{v_L w} \tag{10}$$

$$C_{L'} = C_L \cup C_w \tag{11}$$

$$c_{L'} = c_L + d'_{v_L w} - \sum_{i \in C_w \backslash C_L} \sigma_i \tag{12}$$

$$NG_{L'} = (NG_L \cap N_w) \cup \{w\} \tag{13}$$

where C_w, $w \in O \cup M$, is the set of customers of C that can be covered by w and N_w, $w \in O \cup M$, is the pre-defined ng-sets at service point w.

Propagation rules (8), (9), (10), (13) are commonly used in column generation algorithm for many routing problems. On the contrary, rules (11) and (12) are specific for the MCTP as they depend on the set of customers C. Propagation rule (11) updates the covered customers set of C_L by adding all customers in C that are covered by the service point w. Propagation rule (12) updates the reduced cost by adding, to the cost of label L, the reduced cost of edge $\{v_L, w\}$ and by removing the dual variables of customers covered by w that are not covered by L.

2.2.3. Dominance rule

The following dominance rule can be used to discard labels that cannot generate an optimal ng-route. A label L^1 dominates a label L^2 if the following conditions are satisfied:

$$v_{L^1} = v_{L^2} \tag{14}$$

$$q_{L^1} \leq q_{L^2} \tag{15}$$

$$\tau_{L^1} \leq \tau_{L^2} \tag{16}$$

$$NG_{L^1} \subseteq NG_{L^2} \tag{17}$$

$$c_{L^1} \leq c_{L^2} - \sum_{i \in C_{L^1} \setminus C_{L^2}} \sigma_i \tag{18}$$

and one of the conditions (14)-(18) is strictly satisfied. The first four conditions (14)-(17) ensure that every service point $i \in O \cup M$ that can be attained by propagating label L^2 can also be attained by L^1. Condition (18) states that any propagation of label L^1 has a lower cost of the same propagation of label L^2. The correctness of the dominance conditions (14)-(18) is proved by the following proposition.

Proposition 1. *Given two labels L^1 and L^2 satisfying rules* (14)-(18)*, any extension of L^2 will be dominated by the same extension of L^1.*

Proof. Let us suppose i is a service point accessible for both L^1 and L^2, and i only covers customer j. Let L^1_e (respectively L^2_e) be the extension of L^1 to i (respectively L^2). According to the extension rule (12), the following situations can occur:

- $j \in C_{L^1}$ and $j \in C_{L^2}$. Then, $c_{L^1_e} = c_{L^1} + d'_{v_{L^1} i}$ and $c_{L^2_e} = c_{L^2} + d'_{v_{L^1} i}$. Thus, $c_{L^1_e} \leq c_{L^2_e}$.
- $j \in C_{L^1}$ and $j \notin C_{L^2}$. Then, $c_{L^1_e} = c_{L^1} + d'_{v_{L^1} i}$ and $c_{L^2_e} = c_{L^2} + d'_{v_{L^1} i} - \sigma_j$. However, the rule (18) already ensures that the reduced cost c_{L^1} is lower or equal than reduced cost c_{L^2} minus the dual value $\sigma_j \geq 0$. Thus, $c_{L^1_e} \leq c_{L^2_e}$.
- $j \notin C_{L^1}$ and $j \in C_{L^2}$. Then, $c_{L^1_e} = c_{L^1} + d'_{v_{L^1} i} - \sigma_j$ and $c_{L^2_e} = c_{L^2} + d'_{v_{L^1} i}$. Thus, $c_{L^1_e} \leq c_{L^2_e}$.
- $j \notin C_{L^1}$ and $j \notin C_{L^2}$. Then, $c_{L^1_e} = c_{L^1} + d'_{v_{L^1} i} - \sigma_j$ and $c_{L^2_e} = c_{L^2} + d'_{v_{L^1} i} - \sigma_j$. Thus, $c_{L^1_e} \leq c_{L^2_e}$.

As rule (18) states that the reduced cost $\sigma_i \geq 0$ of every customer i covered by L^1 but not covered by L^2 is subtracted from the reduced cost of c_{L^2}, any extension of L^2 is dominated by the same extension of L^1.

□

2.2.4. Completion bounds

The following completion bounds can be used to discard labels that cannot generate an ng-route with a negative reduced cost. The completion bounds presented in this subsection are the most straightforward way to apply the completion bounds of state-of-the-art methods for solving vehicle routing problems.

A backward ng-path (NG, q, i) is a non-necessarily elementary path $p = (i_{|S_p|}, .., i_1, 0)$ that starts at the service point $i_{|S_p|} = i$, visits a set of service points S_p such that $NG_p = NG$, has a cardinality $q = q_p$, and ends at the depot.

Let $f(NG, q, i)$ be a lower bound on the least-reduced cost of a backward ng-path (NG, q, i). The dynamic programming algorithm to compute these bounds are explained in Baldacci et al. (2011). For all backward ng-path p starting at the service point v_p, of cardinality q_p and of visited service points S_p, the reduced cost of p is bounded as follows:

$$c_p \geq \min_{NG \subseteq S_p \cap N_{v_p}} f(NG, q_p, v_p)$$

Let us introduce $F(S_{p'}, q_{p'}, v_{p'})$ the lower bound on the reduced cost of all ng-routes that can be constructed by extending the forward ng-path p'.

$$F(S_{p'}, q_{p'}, v_{p'}) = \min_{\substack{q_{v_{p'}} \leq q \leq Q - q_{p'} + q_{v_{p'}} \\ NG \cap S_{p'} = v_{p'}}} f(NG, q, v_{p'})$$

Therefore, any extension of a forward ng-path p' has a reduced cost greater than a value δ if:

$$c_{p'} + F(NG_{p'}, q_{p'}, v_{p'}) > \delta$$

By setting $\delta = 0$, and computing the reduced cost of the ng-path with respect to the dual solution $(\lambda, \gamma, \sigma)$ of RMP, we use completion bounds to discard ng-paths that cannot be expanded to give an ng-route with a negative reduced cost.

For the MCTP, the reduced cost depends on the customers covered by a path, but this information cannot be stored during the algorithm for computing completion bounds. Therefore, the cost used for the completion bounds is not the reduced cost, but the cost cf defined as follows:

$$cf_{ij} = d'_{ij} - \sum_{k \in C_i \setminus C_j} \sigma_k \qquad \forall \{i, j\} \in E, i \in O \cup M \cup \{0\} \tag{19}$$

where C_i is the set of customers of C that can be covered by i.

2.2.5. New completion bounds

As previously noticed in 2.2.4, the bound f is not computed with the reduced cost c, but with another cost cf. The difference lies in the number of times the dual variables σ associated to the customers to be covered are deducted from the cost. For a path p, both costs c_p and cf_p are computed by deducting from the cost c_p, the dual variables associated to visited service points. For c_p, the dual variable σ_i of covered customers $i \in C_p$ is subtracted exactly one time. On the contrary, for cf_p, the dual variable for the covered customer i is deducted multiple times, i.e. the number of visited service nodes that can cover i. Therefore, for each path p, $cf_p \leq \overline{c}_p$. Thus, the bound f can be strenghened by computing a new bound g where the dual variables σ are subtracted fewer times than in f.

The idea of the completion bound is to refine F by taking into account, for a forward ng-path p, the set $C_p \subseteq C$ already covered by p. An ideal completion bound $G_{ideal}(S_p, q_p, v_p, C_p)$ for p can be computed as follows. We define a new cost cg^{ω}_{ij} for each arc (i, j), $i, j \in O \cup M \cup \{0\}$ that exclude the dual values associated to a subset $\omega \subseteq C$:

$$cg^{\omega}_{ij} = d'_{ij} - \sum_{k \in C_i \setminus \{C_j \cup \omega\}} \sigma_k = cf_{ij} + \sum_{k \in C_i \cup \omega} \sigma_k \quad \forall \{i, j\} \in E, i \in O \cup M \cup \{0\} \tag{20}$$

For a subset $\omega \subseteq C$, it is possible to compute $g^{\omega}(NG, q, i)$ the least-cost cg^{ω} backward ng-path (NG, q, i) using the same algorithm as before. Then, the ideal completion bound can be expressed as:

$$G_{ideal}(S_p, q_p, v_p, C_p) = \min_{\substack{q_{v_p} \leq q \leq Q - q_p + q_{v_p} \\ NG \cap S_p = v_p}} g^{C_p}(NG, q, v_p)$$

It is important to notice that $\forall \{i, j\} \in E, i \in O \cup M \cup \{0\}$ and $\forall \omega \subseteq C$, $cg^{\omega}_{ij} \geq cf_{ij}$. Therefore, the bound G_{ideal} is better than the bound F as the latter does not consider the vertices already covered by the partial path. It is also the best possible bound as the vertices in C already covered are not considered in the cost.

A problem is that the bound on a backward ng-path must be pre-computed for every subset $\omega \subseteq C$. As the number of subset ω is exponential, it is not possible in practice. A solution is to limit the number of tested subsets ω. To do so, we restrict ω to be a subset of $\Psi \subseteq C$. The set Ψ is composed of the $MAX\_SELECTED$ vertices $i \in C$ with the highest values σ_i. We pre-compute the value g^{ω} of the backward ng-path for all subsets ω of Ψ. Then, the completion bound for a forward ng-path p used in the algorithm is:

$$G(S_p, q_p, v_p, C_p) = \min_{\substack{q_{v_p} \leq q \leq Q - q_p + q_{v_p} \\ NG \cap S_p = v_p}} g^{C_p \cap \Psi}(NG, q, v_p)$$

While G is lower or equal to G_{ideal}, it is still better than the bound F.

2.2.6. Heuristic

A heuristic is used to speed up the first iterations of the column generation process. It consists of limiting the number of labels for each cardinality q and each ending service point i to a maximum value *MAX_LABEL* during the labeling algorithm. When no column with negative reduced cost is found, the value *MAX_LABEL* can be increased. At the last iteration of the pricing problem, the parameter *MAX_LABEL* has to be set equal to $+\infty$ to guarantee that no negative reduced cost columns exist.

2.2.7. Stabilization technique

In general, the set covering problem is highly degenerate, and the column generation procedure is usually affected by convergence issues due to unstable dual variables that highly fluctuate over the iterations. To limit such instability, stabilization methods are usually applied to the master problem. Du Merle et al. (1999) have proposed the Box Stabilization method, which imposes a soft limit on the dual variables that is updated during the resolution and a penalization on the objective if the limit is exceeded. Pigatti et al. (2005) proposed a simplified version of the Box Stabilization of Du Merle et al. (1999).

We adjust the Box Stabilization technique of Pigatti et al. (2005) to obtain the MP defined by the formulation (21)-(30). Let m^-, m^+, o and w be four sets of artificial variables. Such variables have a cost $\overline{\pi}$ and are restricted to be lower or equal than a positive value α.

$$\min z(\overline{\pi}, \alpha) = \sum_{r \in R} d_r x_r + \sum_{i \in M} \overline{\pi}_i (m_i^+ - m_i^-) - \sum_{i \in O} \overline{\pi}_i o_i + \sum_{i \in C} \overline{\pi}_i w_i \tag{21}$$

$$s.t. \sum_{r \in R} a_{ir} x_r - m_i^- + m_i^+ = 1 \qquad i \in M \tag{22}$$

$$\sum_{r \in R} a_{ir} x_r - o_i \leq 1 \qquad i \in O \tag{23}$$

$$\sum_{r \in R} b_{ir} x_r + w_i \geq 1 \qquad i \in C \tag{24}$$

$$\sum_{r \in R} x_r \leq |K| \tag{25}$$

$$0 \leq m_i^- \leq \alpha \qquad i \in M \tag{26}$$

$$0 \leq m_i^+ \leq \alpha \qquad i \in M \tag{27}$$

$$0 \leq o_i \leq \alpha \qquad i \in O \tag{28}$$

$$0 \leq w_i \leq \alpha \qquad i \in C \tag{29}$$

$$x_r \geq 0 \qquad r \in R \tag{30}$$

Let us notice that for all values of $\alpha \geq 0$ and $\overline{\pi}$, $z(\overline{\pi}, \alpha) \leq z$. Thus solving this formulation provides a valid lower bound for formulation (1)-(6).

Let $\mu^- \geq 0$, $\mu^+ \geq 0$, $\theta \geq 0$ and $\omega \geq 0$ be the dual variables of constraints (26), (27),(28) and (29), respectively. The dual of formulation (21)-(30) presents the following new constraints: $\overline{\pi}_i - \mu_i^- \leq \lambda_i \leq \overline{\pi}_i + \mu_i^+$ ($i \in M$), $\overline{\pi}_i - \theta_i \leq \gamma_i \leq 0$ ($i \in O$) and $0 \leq \sigma_i \leq \overline{\pi}_i + \omega_i$ ($i \in C$). These constraints imply that the deviation of the dual variable λ_i ($i \in M$) from the value $\overline{\pi}_i$ ($i \in M$) penalized by $\alpha|\overline{\pi}_i - \lambda_i|$. The penalization process is the same for γ and σ.

The setting of the parameters $\overline{\pi}$ and α during column generation is made in the following way. The coefficient $\overline{\pi}$ changes during the execution according to the dual variables value of constraints (22)-(24), and α is set to a constant value. Let RMP' be the formulation (21)-(30) with a restricted set of columns and $\widetilde{R'}_0$ be the initial set of columns. First, $\overline{\pi}$ is set to 0 and α to 0.001. At the beginning of the k^{th} iteration of the column generation, the value of $\overline{\pi}$ is set to the value of the optimal dual solution of RMP' with the set $\widetilde{R'}_{k-1}$ of columns. When RMP' is solved to optimality for $\alpha = 0.001$ with a set $\widetilde{R'}_n$ of columns, the whole process is run again for $\alpha = 0$ but the initial set of columns is $\widetilde{R'}_n$ and $\overline{\pi}$ is the optimal dual solution of RMP' with the set $\widetilde{R'}_n$ of columns.

2.3. Step 2: Computing an Upper Bound UB

At the end of the *LB* computation, a set $\widetilde{R'} \subseteq R'$ of ng-routes is available. A subset $\widetilde{R'}_{elem}$ of feasible elementary routes can be easily obtained by removing all non-elementary routes from $\widetilde{R'}$. The upper bound UB of formulation (1)-(6) is computed by solving it on the set $\widetilde{R'}_{elem}$ of routes. In the case no upper bound is found by this heuristic, we set UB to a big enough cost.

2.4. Step 3: Enumerating Columns in the Gap

For the third step, we use a mono-directional labeling algorithm. In this algorithm, a label L is a feasible path that starts at the depot, visits the service points S_L, covers the customers C_L, and ends at the service point v_L with a cardinality q_L, a traveling distance τ_L and a reduced cost c_L. The extension rule from a label L to a label L' is defined by the rules (8)-(12) plus the update of the visited service points set (31).

$$S_{L'} = S_L \cup \{v_{L'}\} \tag{31}$$

The dominance rule used to discard unpromising labels during the column enumeration is the following. A label L^1 dominates a label L^2 if the following conditions are satisfied:

$$v_{L^1} = v_{L^2} \tag{32}$$

$$q_{L^1} = q_{L^2} \tag{33}$$

$$\tau_{L^1} \leq \tau_{L^2} \tag{34}$$

$$S_{L^1} = S_{L^2} \tag{35}$$

$$c_{L^1} \leq c_{L^2} \tag{36}$$

and at least one of conditions (32)-(36) is strictly satisfied. The rule (33) is implied by the rule (35), and is used for computational efficiency. Furthermore, the last rule (36) ensures that the reduced cost of L^1 is lower than the one of L^2. We can notice that no resources are included in the pricing algorithm for the non-robust inequalities (4), and therefore they do not affect its performance. Indeed, the rule (35) implies that it does not exist any client $i \in C_{L^1} \backslash C_{L^2}$. So, an extension of the label L^2 could not have a better reduced cost than the same extension of label L^1.

The same completion bounds of the algorithm for the ng-SPPRC explained in Section 2.2.5 are used, but the label is discarded by setting δ to the value of gap equal to $UB - LB$.

At the end of column enumeration, $\overline{R}$ contains all feasible routes with a reduced cost not greater than the gap that have not been discarded by the dominance rule and the completion bounds.

2.5. Step 4: Computing Integer Optimal Solution

Once the column enumeration in the gap has been performed, we solve formulation (1)-(6) on the restricted set of columns $\overline{R}$ with a general-purpose MILP solver.

2.6. Implementation Details

Let LB be the optimal solution of MP, and let UB_1 be the upper bound obtained by the heuristic of step 2. For computational efficiency, if UB_1 is greater than $1.04 * LB$, then we use as upper bound $UB_2 = 1.04 * LB$ at the step 3. If no feasible solution is found with this new gap in Step 4, or if the feasible solution UB_3 has a cost greater than UB_2 the algorithm is launched again with the minimum between UB_1 and UB_3 as upper bound for the step 3.

3. Exact Method for the BOMCTP

In this section, we present the exact method applied to solve the BOMCTP where the objectives are the minimization of the total distance of the routes and the minimization of the maximum coverage distance η. First, we introduce a set-partitioning formulation of the BOMCTP. Then, the ϵ-constraint formulation is introduced, and the ϵ-constraint method is explained.

3.1. Mathematical Formulation

The following formulation of the BOMCTP is an extension of formulation (1)-(6) and has been presented by Sarpong et al. (2013). Let us notice that constraints (3) and (5) are not in the formulation of Sarpong et al. (2013). For an ng-route $r \in R'$, let κ_r be the maximum distance between each customer in C_r and its nearest visited service point in S_r.

$$\min z = \sum_{r \in R} d_r x_r \tag{37}$$

$$\min w = \eta \tag{38}$$

$$s.t.\ (2) - (6) \tag{39}$$

$$\eta \geq \kappa_r x_r \qquad\qquad r \in R \tag{40}$$

The new objective function (38) minimizes the maximum coverage distance of all routes η. Constraints (40) impose η to exceed the maximum coverage distance κ_r of each route r. We suppose the coverage distance to be integer.

The formulation (37)-(40) works on the set of feasible solutions $\mathcal{X}$, e.g., combination of routes of R that respect all the constraints. The image of $\mathcal{X}$ under the objective function mapping represents the objective space $\mathcal{Y} := (z(\mathcal{X}), w(\mathcal{X}))$. The aim of the BOMCTP is to find all solutions $x \in \mathcal{X}$ which are an interesting compromise between the objectives z and w. These solutions are the image of points $y \in \mathcal{Y}$, $y := (z(x), w(x))$, that are consistent with the following definition, and are called non-dominated points.

Definition 1. *A point $y \in \mathcal{Y}$, image of solution $a \in \mathcal{X}$, is said to be a non-dominated point if $\nexists\ b \in \mathcal{X}$, $b \neq a$ such that $z(b) < z(a)$ and $w(b) < w(a)$.*

The Pareto Front obtained by solving the formulation (37)-(40) is composed of all non-dominated points. However, as the same point $y \in \mathcal{Y}$ can be associated with several different solutions in $\mathcal{X}$, the

number of efficient solutions can be larger than the number of non-dominated points. In this paper, we only search for one solution associated to each non-dominated point.

3.2. ϵ-Constraint Method

The ϵ-constraint is an efficient method for solving bi-objective problems introduced by Haimes (1971). The method consists of modifying the formulation of the BOMCTP into a formulation with one objective, and solves a sequence of mono-objective problems.

First, the formulation (37)-(40) is modified into the following ϵ-constraint formulation. A constant ϵ is introduced to constrain one of the costs to be lower than the constant. For a specific ϵ, only one non-dominated point of formulation (37)-(40) is found. Therefore, to find the complete Pareto front, the ϵ value has to vary.

$$\min z = \sum_{r \in R} d_r x_r \tag{41}$$

$$s.t. \sum_{r \in R} a_{ir} x_r = 1 \qquad i \in M \tag{42}$$

$$\sum_{r \in R} a_{ir} x_r \leq 1 \qquad i \in O \tag{43}$$

$$\sum_{r \in R} b_{ir} x_r \geq 1 \qquad i \in C \tag{44}$$

$$\sum_{r \in R} x_r \leq |K| \tag{45}$$

$$\epsilon \geq \kappa_r x_r \qquad r \in R \tag{46}$$

$$x_r \in \{0, 1\} \qquad r \in R \tag{47}$$

As noticed in Sarpong et al. (2013), the structure of a bi-objective problem with a min-max objective is specific, and the ϵ-constraint (46) can be discarded in the formulation and replaced by a requirement on the definition of a feasible route. Therefore, we use the formulation (1)-(6) with a maximum coverage distance η set to ϵ to solve the BOMCTP.

When a non-dominated point OPT is found, the value ϵ can be set to $\eta(\text{OPT}) - 1$, with $\eta(\text{OPT})$ the maximum coverage distance of the solution associated to OPT, as η is an integer coefficient due to the fact that the coverage distance are supposed integer. However, ϵ can be decreased by more than one as explained in Jozefowiez et al. (2007) and in the following *compute_next_covering* algorithm. The

ϵ-constraint method can be summed up as in Algorithm 1, and the global algorithm uses three other ones:

- *solution_exists*(ϵ) checks if a solution that respects all the constraints is possible for $\eta = \epsilon$. For this, it is sufficient to check if all customers in C are covered by at least one optional service point of O.
- *solve_MP*(ϵ) solves the mono-objective formulation (1)-(6) with a maximum coverage distance set to ϵ and returns the optimal primal solution OPT.
- *compute_next_covering* returns the maximal value of η for which the current solution OPT is not valid anymore; i.e., returns the maximal value κ_r ($\forall r |$ r selected by OPT).

Result: P the set of non-dominated points of BOMCTP

$\epsilon = +\infty$;

while *solution_exists*(ϵ) **do**

  OPT = *solve_MP*(ϵ);

  $P \leftarrow P \cup \{OPT\}$;

  ϵ = *compute_next_covering*

end

Return P;

Algorithm 1: *reduced_ϵ_constraint_method(G)*

4. Computational Results

In this section, we present the computational results of the exact methods presented in Sections 2 and 3 for the MCTP and the BOMCTP. The experiments have been conducted on a Xeon E5-2695 processor with a 2.30GHz CPU on a single thread. The implementation is in C++, and the linear problem (21)-(30) and the integer problem (1)-(6) are solved with Gurobi 7.1.

The following parameter setting has been used: $\Delta = 8$ (see Section 2.1.1), *MAX_SELECTED* = 3 (see Section 2.1.5), and *MAX_LABEL*=4, 30, and $+\infty$ (see Section 2.1.6).

The time limit is 2 hours for the mono-objective method, 3 hours for the computation of the lower bound set of the BOMCTP, and 6 hours for the exact bi-objective method. If a method does not converge in the time limit, the corresponding line is noted with a dash (-). The best methods in terms of CPU times and gap are noted in bold.

4.1. Mono-Objective Version

The instances of the MCTP are the ones used in Jozefowiez (2014) and Ha et al. (2013). They are named *X-T-n-W-p* where *X* is the name of the Krolak instance from which the instance is derived, *T* is the size of the mandatory service point set plus the depot, *n* is the size of the service point set plus the depot, *W* is the size of the customer set and *p* is the maximum number of visited service points allowed in a route. In total 96 instances are tested.

The state-of-the-art exact method for the MCTP is presented in the paper of Ha et al. (2013), that is why we compare our results with them in the following tables. As the CPU used by Hà et al. is different, we use the passmark CPU score[1] with a single thread to construct a fair coefficient. They use a 2.4-GHz Intel Core2 Duo CPU and we use a 2.1-GHz Xeon E5-2695 v4 CPU. They do not specify exactly the model of their microprocessor, and the CPU Single Thread Rating of the 2.4-GHz Intel Core2 Duo CPU varies between 842 and 991. Thus, we compare with the less efficient Intel Core2 Duo processor available. So, the CPU Single Thread Rating of the CPU of Hà et al. is 842 and ours is 1628. We report in Tables 2, 3 and 4 their times divided by 1.94 in column *Time_c* and their original times in column *Time_o*.

In these three tables, the following column headings are: the CPU time in seconds (*Time*), the optimal solution of each instance (UB), the lower bound of the MCTP (LB), and the gap between the lower bound LB and the optimal solution UB in percentage (*Gap*). If no optimal solution is found, the gap is computed with a valid upper bound. The gap is computed as follows:

$$Gap = 100 - \frac{z(LB)*100}{z(UB)} \ (\%);$$

The computational results show that our method is strictly more efficient than the state-of-the-art method on 82 instances out of 96. Our method is strictly less efficient on 7 instances. Among these 7 instances, the difference of time is higher than 5 seconds only on D1-1-50-50-8. It could be explained as column generation is less efficient on vehicle routing problems with long routes. For instances with $|V| = 200$, our method outperforms the one of Hà et al. We close 7 open instances. On the 6 instances that remain unsolved, the lower bound we find is clearly better than the one of the method of Hà et al.

We can notice that the difficulty of the instance increases with the maximum cardinality of the route. The number of service points also plays an important part in the difficulty as well as the number of

[1]Data available in the website `https://www.cpubenchmark.net/cpu_list.php` (*28/10/2019*)

Table 2: Computational results of our method and method of Ha et al. (2013) on instances with $|V| = 100$ and $|M| = 0$

		Ha et al. (2013)				Our method		
Instance	UB	LB	Gap	Time_o	Time_c	LB	Gap	Time
A1-1-25-75-4	8479.0	8218.0	3.1	1.1	0.6	8479.0	**0.0**	**0.1**
A1-1-25-75-5	8479.0	7521.1	11.3	3.3	1.7	8479.0	**0.0**	**0.2**
A1-1-25-75-6	8479.0	7411.3	12.6	3.3	1.7	8402.3	**0.9**	**0.6**
A1-1-25-75-8	7985.0	6340.3	20.6	20.1	10.4	7985.0	**0.0**	**0.6**
A1-1-50-50-4	10271.0	9346.0	9.0	9.9	5.1	10271.0	**0.0**	**0.6**
A1-1-50-50-5	9220.0	7925.4	14.0	12.4	6.4	9220.0	**0.0**	**0.6**
A1-1-50-50-6	9130.0	7186.3	21.3	24.8	12.8	9130.0	**0.0**	**3.1**
A1-1-50-50-8	9130.0	6325.9	30.7	203.9	105.1	9031.0	**1.1**	**24.8**
B1-1-25-75-4	7146.0	6563.8	8.1	1.8	0.9	7146.0	**0.0**	**0.2**
B1-1-25-75-5	6901.0	5821.7	15.6	3.2	1.6	6782.0	**1.7**	**0.3**
B1-1-25-75-6	6450.0	5009.6	22.3	4.3	2.2	6450.0	**0.0**	**0.5**
B1-1-25-75-8	6450.0	4464.5	30.8	10.9	5.6	6450.0	**0.0**	**2.4**
B1-1-50-50-4	10107.0	8706.1	13.9	16.6	8.6	10107.0	**0.0**	**0.6**
B1-1-50-50-5	9723.0	7750.3	20.3	84.1	43.4	9723.0	**0.0**	**1.0**
B1-1-50-50-6	9382.0	7193.3	23.3	162.2	83.6	9273.0	**1.2**	**3.4**
B1-1-50-50-8	8348.0	5744.5	31.2	76.1	39.2	8348.0	**0.0**	**10.1**
C1-1-25-75-4	6161.0	5265.9	14.5	2.8	1.4	6161.0	**0.0**	**0.2**
C1-1-25-75-5	6161.0	4946.5	19.7	5.8	3.0	6161.0	**0.0**	**0.3**
C1-1-25-75-6	6161.0	4678.2	24.1	7.7	4.0	6161.0	**0.0**	**0.5**
C1-1-25-75-8	6161.0	4279.7	30.5	9.4	4.8	6161.0	**0.0**	**1.6**
C1-1-50-50-4	11372.0	10462.3	8.0	8.1	4.2	11042.0	**2.9**	**0.9**
C1-1-50-50-5	9900.0	8897.0	10.1	13.3	6.9	9900.0	**0.0**	**0.7**
C1-1-50-50-6	9895.0	8210.6	17.0	56.9	29.3	9719.0	**1.8**	**3.1**
C1-1-50-50-8	8699.0	7479.8	14.0	8.5	**4.4**	8684.0	**0.2**	21.7
D1-1-25-75-4	7671.0	7471.4	2.6	1.0	0.5	7671.0	**0.0**	**0.2**
D1-1-25-75-5	7465.0	6630.6	11.2	5.4	2.8	7363.3	**1.4**	**0.6**
D1-1-25-75-6	6651.0	5764.1	13.3	3.8	2.0	6651.0	**0.0**	**0.3**
D1-1-25-75-8	6651.0	5187.7	22.0	12.9	6.6	6651.0	**0.0**	**1.3**
D1-1-50-50-4	11606.0	10704.4	7.8	9.3	4.8	11606.0	**0.0**	**0.7**
D1-1-50-50-5	10770.0	9139.6	15.1	29.3	15.1	10770.0	**0.0**	**0.9**
D1-1-50-50-6	10525.0	8264.9	21.5	281.3	145.0	10197.0	**3.1**	**4.0**
D1-1-50-50-8	9361.0	6954.8	25.7	110.6	**57.0**	9341.4	**0.2**	121.2
Mean			17.0	37.6	19.4		**0.5**	**6.5**
Closed instance			32/32				32/32	

mandatory service points. The number of customers does not seem as important as the other parameters.

To evaluate the efficiency of the new completion bound g introduced in Section 2.2.5, we launch our method with the state-of-the-art completion bound f and with the new completion bound g. The results are presented in Tables 5 and 6. In these tables, the following column headings are: the CPU time in seconds (*Time*), and the number of labels accepted by the completion bounds during the column generation over the last iterations to compute the lower bound of the MCTP (*Allowed path*). The last iterations corresponds to the subproblem solved with MAX_LABEL= $+\infty$ (section 2.2.6). The smaller the number of allowed paths, the stronger the completion bound is.

Table 3: Computational results of our method and method of Ha et al. (2013) on instances with $|V| = 100$ and $|M| > 0$

		Ha et al. (2013)				Our method		
Instance	UB	LB	Gap	Time_o	Time_c	LB	Gap	Time
A1-5-25-75-4	10827.0	9466.4	12.6	9.5	4.9	9894.8	**8.6**	**0.3**
A1-5-25-75-5	8659.0	8659.0	**0.0**	0.1	**0.1**	8659.0	**0.0**	0.2
A1-5-25-75-6	8659.0	8576.6	1.0	6.9	3.6	8659.0	**0.0**	**0.3**
A1-5-25-75-8	8265.0	7398.2	10.5	4.2	2.2	8265.0	**0.0**	**0.5**
A1-10-50-50-4	17953.0	16223.0	9.6	4828.6	2489.0	16981.0	**5.4**	**11.1**
A1-10-50-50-5	15440.0	14317.2	7.3	173.6	89.5	15051.9	**2.5**	**2.0**
A1-10-50-50-6	14064.0	12769.9	9.2	1586.2	817.6	13831.5	**1.7**	**12.3**
A1-10-50-50-8	-	11280.3	17.7	-	-	12050.0	**12.1**	-
B1-5-25-75-4	9465.0	9465.0	**0.0**	0.2	**0.1**	9465.0	**0.0**	0.2
B1-5-25-75-5	9460.0	8713.2	7.9	5.7	2.9	9293.3	**1.8**	**0.3**
B1-5-25-75-6	9148.0	8291.1	9.4	18.1	9.3	9011.0	**1.5**	**0.5**
B1-5-25-75-8	8306.0	7193.7	13.4	8.9	4.6	8306.0	**0.0**	**1.1**
B1-10-50-50-4	15209.0	14106.9	7.2	127.6	65.8	14878.3	**2.2**	**0.6**
B1-10-50-50-5	13535.0	12125.2	10.4	149.2	76.9	13089.5	**3.3**	**2.3**
B1-10-50-50-6	12067.0	10815.9	10.4	104.7	54.0	11791.0	**2.3**	**1.7**
B1-10-50-50-8	10344.0	9343.0	9.7	32.3	16.6	10344.0	**0.0**	**1.9**
C1-5-25-75-4	9898.0	9795.5	1.0	0.4	**0.2**	9898.0	**0.0**	**0.2**
C1-5-25-75-5	9707.0	9205.3	5.2	3.0	1.5	9314.4	**4.0**	**0.4**
C1-5-25-75-6	9321.0	8688.0	**6.8**	4.2	**2.2**	8534.7	8.4	6.1
C1-5-25-75-8	7474.0	7474.0	**0.0**	9.4	4.8	7474.0	**0.0**	**0.9**
C1-10-50-50-4	18212.0	17217.0	5.5	164.4	84.7	17898.0	**1.7**	**0.9**
C1-10-50-50-5	16362.0	15583.4	4.8	126.8	65.4	15697.8	**4.1**	**17.2**
C1-10-50-50-6	14749.0	13932.4	5.5	240.4	123.9	14224.2	**3.6**	**11.2**
C1-10-50-50-8	12394.0	12000.6	3.2	5.6	2.9	12394.0	**0.0**	**2.7**
D1-5-25-75-4	11820.0	11241.8	4.9	1.7	0.9	11783.0	**0.3**	**0.4**
D1-5-25-75-5	10982.0	9870.1	10.1	16.7	**8.6**	10465.3	**4.7**	13.1
D1-5-25-75-6	9669.0	8976.1	7.2	3.4	1.8	9517.8	**1.6**	**0.6**
D1-5-25-75-8	8200.0	7516.4	8.3	1.3	**0.7**	8200.0	**0.0**	1.2
D1-10-50-50-4	20982.0	20346.9	3.0	10.9	5.6	20982.0	**0.0**	**0.8**
D1-10-50-50-5	18576.0	9139.6	50.8	393.5	202.8	17984.2	**3.2**	**58.2**
D1-10-50-50-6	16330.0	15073.8	7.7	116.1	59.8	16156.0	**1.1**	**3.2**
D1-10-50-50-8	14204.0	12613.5	11.2	248.4	128.0	13909.0	**2.1**	**6.9**
Mean			8.5	271.0	139.7		**2.4**	**5.5**
Closed instance			31/32				31/32	

The computational results show that the bound g is strictly more efficient in terms of computing time on 43 instances out of 96. The bound f is strictly more efficient on 31 instances out of 96. The mean time for all instances is 139.7 and 153.8 for the bound g and f, respectively.

We can also notice that the number of accepted paths in the sub-problem after the completion bounds is always higher for the bound f as the g ones prune more labels in theory. For example, the instance A1-1-50-50-8 shows that the number of accepted paths is nearly four times superior with the bound f and the impact in the CPU time is proportional. However, on the instance C1-10-50-50-5, few paths are allowed with both bounds and therefore, the time to compute the bound has a negative impact on the CPU time for g one as it is more expensive to compute.

Table 4: Computational results of our method and method of Ha et al. (2013) on instances with $|V| = 200$

		Ha et al. (2013)				Our method		
Instance	UB	LB	Gap	Time_o	Time_c	LB	Gap	Time
A2-1-50-150-4	11550.0	10437.8	9.6	82.2	42.4	11508.2	**0.4**	**1.8**
A2-1-50-150-5	10407.0	9168.5	11.9	340.6	175.6	10407.0	**0.0**	**2.8**
A2-1-50-150-6	10068.0	8527.6	15.3	1075.8	554.5	10068.0	**0.0**	**3.6**
A2-1-50-150-8	8896.0	7169.3	19.4	153.4	79.1	8896.0	**0.0**	**23.9**
A2-1-100-100-4	11885.0	9675.9	18.6	4593.9	2368.0	11512.2	**3.1**	**20.8**
A2-1-100-100-5	10234.0	8212.1	19.8	1440.1	742.3	10234.0	**0.0**	**13.9**
A2-1-100-100-6	10020.0	7162.9	28.5	-	-	10020.0	**0.0**	**29.5**
A2-1-100-100-8	9093.0	5883.0	35.3	-	-	9093.0	**0.0**	**826.8**
A2-10-50-150-4	17083.0	16049.6	6.0	1257.0	647.9	16761.0	**1.9**	**1.8**
A2-10-50-150-5	14977.0	13905.0	7.2	494.7	255.0	14803.5	**1.2**	**4.5**
A2-10-50-150-6	13894.0	12959.8	6.7	978.9	504.6	13453.5	**3.2**	**40.8**
A2-10-50-150-8	11942.0	11207.4	6.2	280.2	144.4	11942.0	**0.0**	**9.8**
A2-20-100-100-4	26594.0	25042.6	5.8	-	-	26174.0	**1.6**	**10.2**
A2-20-100-100-5	23419.0	21331.4	8.9	-	-	22715.5	**3.0**	**4559.9**
A2-20-100-100-6	-	18775.1	11.4	-	-	20372.0	**3.8**	-
A2-20-100-100-8	-	15552.1	17.0	-	-	17511.1	**6.6**	-
B2-1-50-150-4	11175.0	9573.0	14.3	166.0	85.6	11175.0	**0.0**	**1.5**
B2-1-50-150-5	10502.0	8245.4	21.5	1114.7	574.6	10339.0	**1.6**	**2.8**
B2-1-50-150-6	9799.0	7604.1	22.4	1274.0	656.7	9799.0	**0.0**	**3.5**
B2-1-50-150-8	8846.0	6213.4	29.8	166.0	85.6	8846.0	**0.0**	**8.7**
B2-1-100-100-4	18370.0	16748.9	8.8	6615.0	3409.8	17864.2	**2.8**	**314.4**
B2-1-100-100-5	15876.0	14214.5	10.5	1472.0	758.8	15838.5	**0.2**	**11.7**
B2-1-100-100-6	14867.0	12394.7	16.6	-	-	14531.2	**2.3**	**107.7**
B2-1-100-100-8	-	10366.0	25.7	-	-	12831.0	**8.1**	-
B2-10-50-150-4	16667.0	15125.6	9.2	5972.5	3078.6	15675.3	**6.0**	**9.7**
B2-10-50-150-5	14188.0	13574.2	4.3	124.2	64.0	13967.5	**1.6**	**4.5**
B2-10-50-150-6	12954.0	11947.6	7.8	773.6	398.8	12561.5	**3.0**	**4.4**
B2-10-50-150-8	11495.0	10078.3	12.3	732.7	377.7	11277.2	**1.9**	**13.2**
B2-20-100-100-4	34062.0	32913.3	3.4	-	-	33790.0	**0.8**	**9.2**
B2-20-100-100-5	29405.0	27494.1	6.5	-	-	28769.8	**2.2**	**4961.8**
B2-20-100-100-6	-	23954.7	8.2	-	-	25262.7	**2.7**	-
B2-20-100-100-8	-	19488.6	14.8	-	-	21441.0	**6.3**	-
Mean			13.9	1455.4	750.2		**2.0**	**393.1**
Closed instance			20/32				27/32	

4.2. Bi-Objective Version

As far as we know, there is no exact method for the BOMCTP. The instances used are the MCTP instances without a fixed covering distance η. The results are reported in Table 7. This table presents a column $|\mathcal{Y}|$ for the number of non-dominated points, and a column *Time* for the CPU time in seconds to obtain the complete Pareto front.

We can notice that 62 instances out of 96 are solved to optimality in 6 hours. The number of non-dominated points is quite high: between 12 and 90, with an average of 40. This means that this variant of the BOMCTP offers a compromise between the two objectives. For example, the instance D1-1-25-75-6 contains 29 non-dominated points, and the two extreme points have a cost of 730 and 11047 and a

Table 5: Comparison of the completion bounds f and g on instances with $|V| = 100$ and $|M| \geq 0$

	State-of-the-art completion bound f		New completion bound g			State-of-the-art completion bound f		New completion bound g	
Instance	Allowed path	Time	Allowed path	Time	Instance	Allowed path	Time	Allowed path	Time
A1-1-25-75-4	498	**0.1**	88	**0.1**	A1-5-25-75-4	166	**0.3**	80	**0.3**
A1-1-25-75-5	2198	**0.2**	1522	**0.2**	A1-5-25-75-5	274	**0.2**	50	**0.2**
A1-1-25-75-6	11060	1.0	5088	**0.6**	A1-5-25-75-6	2126	**0.3**	483	**0.3**
A1-1-25-75-8	65634	1.9	13872	**0.6**	A1-5-25-75-8	18655	0.7	4430	**0.5**
A1-1-50-50-4	270	**0.4**	130	0.6	A1-10-50-50-4	346	**10.9**	158	11.1
A1-1-50-50-5	8446	0.8	1130	**0.6**	A1-10-50-50-5	702	**1.5**	368	2.0
A1-1-50-50-6	74643	**2.7**	56142	3.1	A1-10-50-50-6	4218	12.7	1644	**12.3**
A1-1-50-50-8	228512	130.1	69878	**24.8**	A1-10-50-50-8	-	-	-	-
B1-1-25-75-4	608	**0.1**	250	0.2	B1-5-25-75-4	66	**0.1**	66	0.2
B1-1-25-75-5	4584	0.4	1262	**0.3**	B1-5-25-75-5	3632	**0.3**	857	**0.3**
B1-1-25-75-6	14200	0.6	3600	**0.5**	B1-5-25-75-6	13778	0.9	3800	**0.5**
B1-1-25-75-8	384250	11.3	52298	**2.4**	B1-5-25-75-8	33277	**1.1**	19655	**1.1**
B1-1-50-50-4	934	**0.6**	374	**0.6**	B1-10-50-50-4	340	0.7	290	**0.6**
B1-1-50-50-5	5405	0.8	4379	**1.0**	B1-10-50-50-5	710	**2.1**	604	2.3
B1-1-50-50-6	32358	7.6	13144	**3.4**	B1-10-50-50-6	866	2.0	686	**1.7**
B1-1-50-50-8	410434	10.9	183948	**10.1**	B1-10-50-50-8	8770	2.3	7926	**1.9**
C1-1-25-75-4	358	**0.2**	222	**0.2**	C1-5-25-75-4	214	**0.2**	96	**0.2**
C1-1-25-75-5	15273	0.5	1365	**0.3**	C1-5-25-75-5	580	**0.4**	492	**0.4**
C1-1-25-75-6	47721	1.6	8404	**0.5**	C1-5-25-75-6	308	**5.2**	162	6.1
C1-1-25-75-8	483856	13.3	48235	**1.6**	C1-5-25-75-8	2342	**0.8**	860	0.9
C1-1-50-50-4	1522	**0.8**	1294	0.9	C1-10-50-50-4	242	**0.8**	236	0.9
C1-1-50-50-5	6074	1.0	410	**0.7**	C1-10-50-50-5	1635	**15.5**	1552	17.2
C1-1-50-50-6	84814	14.5	17333	**3.1**	C1-10-50-50-6	4668	**10.2**	3392	11.2
C1-1-50-50-8	257838	32.9	142493	**21.7**	C1-10-50-50-8	7528	**2.7**	3684	**2.7**
D1-1-25-75-4	674	**0.2**	214	**0.2**	D1-5-25-75-4	182	**0.4**	104	**0.4**
D1-1-25-75-5	3620	**0.5**	2502	0.6	D1-5-25-75-5	410	**11.3**	204	13.1
D1-1-25-75-6	9244	0.6	1058	**0.3**	D1-5-25-75-6	1678	**0.5**	788	0.6
D1-1-25-75-8	447168	7.6	38186	**1.3**	D1-5-25-75-8	21128	1.3	18012	**1.2**
D1-1-50-50-4	604	**0.6**	418	0.7	D1-10-50-50-4	428	**0.7**	224	0.8
D1-1-50-50-5	14524	1.0	2726	**0.9**	D1-10-50-50-5	414	**41.8**	278	58.2
D1-1-50-50-6	9672	**4.0**	6132	**4.0**	D1-10-50-50-6	1646	**3.0**	1026	3.2
D1-1-50-50-8	596132	409.6	203502	**121.2**	D1-10-50-50-8	18281	10.8	7140	**6.9**
Mean	100723	20.6	27550	**6.5**	Mean	4826	**4.6**	2560	5.1

maximum coverage distance of 2934 and 658 respectively. The optimal solution of the MCTP found for the instance D1-1-25-75-6 corresponds to a non-dominated point of cost 6651 and of maximum coverage distance 931 situated in the middle of the Pareto front.

We can notice that the value of the maximum cardinality of a route $\overline{n}$ does not influence the number of non-dominated points. On the contrary, the higher the number of mandatory service points, the lower the number of non-dominated points as shown by B1-1-25-75-8 and B1-5-25-75-8 which have 36 and 19 non-dominated points, respectively. The higher the number of optional service points, the higher the number of non-dominated points as shown by A1-1-25-75-4 and A1-1-50-50-4 which have 30 and 50 non-dominated points, respectively.

Table 6: Comparison of the completion bounds f and g on instances with $|V| = 200$

	State-of-the-art completion bound f		New completion bound g			State-of-the-art completion bound f		New completion bound g	
Instance	Allowed path	Time	Allowed path	Time	Instance	Allowed path	Time	Allowed path	Time
A2-1-50-150-4	1462	2.6	1000	**1.8**	B2-1-50-150-4	692	2.1	668	**1.5**
A2-1-50-150-5	27693	5.4	10452	**2.8**	B2-1-50-150-5	3016	3.6	1500	**2.8**
A2-1-50-150-6	104062	7.4	18681	**3.6**	B2-1-50-150-6	13672	**3.5**	6538	**3.5**
A2-1-50-150-8	849383	43.2	278680	**23.9**	B2-1-50-150-8	84521	**8.5**	36993	8.7
A2-1-100-100-4	4182	**19.1**	2984	20.8	B2-1-100-100-4	1450	**269.3**	1198	314.4
A2-1-100-100-5	88582	**12.0**	71965	13.9	B2-1-100-100-5	4710	**8.4**	4582	11.7
A2-1-100-100-6	704792	80.2	188834	**29.5**	B2-1-100-100-6	32613	149.6	31614	**107.7**
A2-1-100-100-8	10610182	**624.2**	7795891	826.8	B2-1-100-100-8	-	-	-	-
A2-10-50-150-4	216	2.4	186	**1.8**	B2-10-50-150-4	120	12.4	76	**9.7**
A2-10-50-150-5	680	**2.8**	666	4.5	B2-10-50-150-5	1433	5.9	1223	**4.5**
A2-10-50-150-6	3677	**36.0**	2848	40.8	B2-10-50-150-6	2074	3.3	1860	**4.4**
A2-10-50-150-8	74259	15.5	45510	**9.8**	B2-10-50-150-8	16382	**9.1**	12242	13.2
A2-20-100-100-4	520	**9.2**	410	10.2	B2-20-100-100-4	700	12.5	666	**9.2**
A2-20-100-100-5	2315	5472.0	1771	**4559.9**	B2-20-100-100-5	1970	**4954.3**	1702	4961.8
A2-20-100-100-6	-	-	-	-	B2-20-100-100-6	-	-	-	-
A2-20-100-100-8	-	-	-	-	B2-20-100-100-8	-	-	-	-
					Mean	467976	436.1	315583	**407.5**

The lower bound sets of the BOMCTP have been studied in Artigues et al. (2018). Table 8 compares the tightness and the efficiency of the lower bound sets obtained by our method and the method from the literature, called SOGA, presented in Artigues et al. (2018). Both algorithms are run on the same CPU, but the method SOGA solves the integer and linear problems with CPLEX 12.9.

The efficiency is evaluated in terms of CPU time used to compute the lower bound sets in column *Time*. The tightness is determined with two values: *Pt* represents the number of points in the lower bound set, and *Hv* represents the hypervolume metric. The hypervolume is the percentage of area in the objective space covered by the lower bound sets compared to an ideal point. The figure 2 illustrates this measure.

In this example, the point N represents a nadir point that is dominated by every non-dominated feasible points of the BOMCTP. This means that the exact Pareto front is contained in the rectangle formed by the points I and N. The points A, B, C and D and the lines between them represent the lower bound set. The hypervolume measure of the lower bound set is equal to the gray area divided by the area of the rectangle formed by the points I and N. It is straightforward that the smaller the measure, the better the lower bound set. For the hypervolume measure of the two methods, the same point N is used for similar instances.

Table 7: Computational results of our bi-objective method

Instance	$\|\mathcal{Y}\|$	Time	Instance	$\|\mathcal{Y}\|$	Time	Instance	$\|\mathcal{Y}\|$	Time	Instance	$\|\mathcal{Y}\|$	Time
A1-1-25-75-4	30	5.4	A1-10-50-50-5	47	20779.2	C1-1-25-75-8	36	1776.2	D1-10-50-50-4	45	5600.2
A1-1-25-75-5	32	5.3	B1-5-25-75-4	21	5.2	C1-1-50-50-4	61	61.0	A2-1-50-150-4	58	275.5
A1-1-25-75-6	32	11.6	B1-5-25-75-5	20	5.4	C1-1-50-50-5	62	431.1	A2-1-50-150-5	52	215.5
A1-1-25-75-8	31	26.3	B1-5-25-75-6	22	20.8	D1-1-25-75-4	29	3.7	A2-1-50-150-6	53	1835.6
A1-1-50-50-4	50	30.4	B1-5-25-75-8	19	1019.2	D1-1-25-75-5	29	6.1	A2-1-50-150-8	55	15240.3
A1-1-50-50-5	49	1007.3	B1-10-50-50-4	33	23.5	D1-1-25-75-6	29	9.4	A2-1-100-100-4	90	13895.3
A1-1-50-50-6	49	1191.4	B1-10-50-50-5	35	357.8	D1-1-25-75-8	29	39.6	A2-1-100-100-5	81	2247.4
A1-1-50-50-8	51	617.2	B1-10-50-50-6	29	6335.4	D1-1-50-50-4	62	47.5	A2-10-50-150-4	47	668.5
B1-1-25-75-4	34	5.3	C1-5-25-75-4	14	2.7	D1-1-50-50-5	65	62.3	A2-10-50-150-5	38	6412.3
B1-1-25-75-5	37	7.1	C1-5-25-75-5	15	4.5	D1-1-50-50-6	63	922.0	B2-1-50-150-4	57	1458.0
B1-1-25-75-6	37	11.7	C1-5-25-75-6	14	16.5	A1-5-25-75-4	24	6.4	B2-1-50-150-5	53	2818.7
B1-1-25-75-8	36	38.9	C1-5-25-75-8	12	6.1	A1-5-25-75-5	21	8.9	B2-1-50-150-6	51	309.8
B1-1-50-50-4	58	1021.0	C1-10-50-50-4	58	1264.2	A1-5-25-75-6	18	20.5	B2-1-50-150-8	56	4741.1
C1-1-25-75-4	39	8.8	C1-10-50-50-5	47	9625.0	A1-5-25-75-8	19	11.4	B2-10-50-150-4	37	211.8
C1-1-25-75-5	38	8.8	D1-5-25-75-4	27	16.7	A1-10-50-50-4	39	1566.8	B2-10-50-150-6	43	1530.0
C1-1-25-75-6	38	35.2	D1-5-25-75-5	25	1294.9						

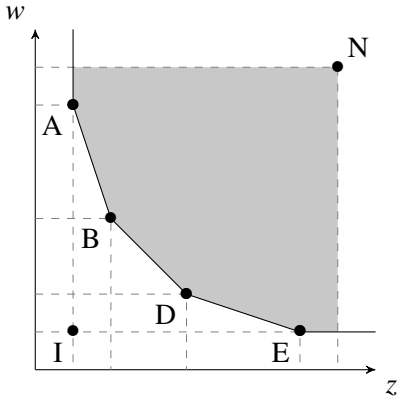

Figure 2: Example of a lower bound set for a bi-objective problem

The results show that our method computes tighter lower bound sets for all instances. The lower bound sets computed with our method generates more lower bound points, and it could explain both a part of the preciseness of our lower bound sets and the difference of time between the two methods. For instance, in *A*2-1-100-100-5, our method computes 113 points in 2022.5 seconds for an hypervolume of 84.5%, and the SOGA method computes 63 points in 1037.2 seconds for an hypervolume of 85.1%. However, for some instances, our method is still faster with more points computed. For instance, in *D*1-1-50-50-8, the SOGA method computes 55 points in 2104.0 seconds, and ours computes 80 points in 1546.9 seconds.

Table 8: Computational results of our bi-objective method and comparison with the method SOGA of Artigues et al. (2018)

	SOGA			Our method				SOGA			Our method		
Instance	$\|ND\|$	Hv	Time	$\|ND\|$	Hv	Time	Instance	$\|ND\|$	Hv	Time	$\|ND\|$	Hv	Time
A1-1-25-75-4	34	83.0	**5.4**	48	**82.8**	5.7	B1-1-25-75-4	40	84.6	10.6	60	**84.5**	**9.0**
A1-1-25-75-5	38	**83.2**	**7.0**	41	**83.2**	8.4	B1-1-25-75-5	39	**84.9**	16.6	42	**84.9**	**9.0**
A1-1-25-75-6	33	**83.3**	**10.6**	46	**83.3**	15.4	B1-1-25-75-6	33	85.2	**14.9**	44	**85.1**	16.2
A1-1-25-75-8	31	83.6	**7.6**	34	**83.5**	36.6	B1-1-25-75-8	33	85.5	**15.6**	41	**85.4**	52.8
A1-1-50-50-4	48	87.7	72.2	74	**87.5**	**34.9**	B1-1-50-50-4	57	86.6	174.0	75	**86.5**	**31.7**
A1-1-50-50-5	50	**88.2**	148.5	67	**88.2**	**45.5**	B1-1-50-50-5	58	**87.0**	238.4	72	**87.0**	**56.6**
A1-1-50-50-6	45	88.6	421.0	58	**88.5**	**82.1**	B1-1-50-50-6	53	**87.3**	946.2	68	**87.3**	**97.0**
A1-1-50-50-8	48	88.9	569.9	58	**88.8**	**351.1**	B1-1-50-50-8	47	87.7	**858.6**	60	**87.6**	1749.7
A2-1-50-150-4	71	84.0	260.7	137	**83.9**	**181.5**	B2-1-50-150-4	48	85.6	147.5	127	**84.9**	**134.2**
A2-1-50-150-5	63	84.8	338.1	77	**84.7**	**199.6**	B2-1-50-150-5	51	85.7	1103.4	91	**85.5**	**191.9**
A2-1-50-150-6	55	85.2	603.2	96	**85.1**	**504.2**	B2-1-50-150-6	55	85.9	1793.5	84	**85.8**	**370.6**
A2-1-50-150-8	54	**85.5**	**835.6**	72	**85.5**	1655.1	B2-1-50-150-8	60	**86.2**	**4642.5**	80	**86.2**	7600.1
A2-1-100-100-4	71	84.1	1812.8	131	**83.8**	**971.0**	B2-1-100-100-4	84	86.2	10522.5	169	**85.8**	**901.8**
A2-1-100-100-5	63	85.1	**1037.2**	113	**84.5**	2022.5	B2-1-100-100-5	-	-	-	162	**86.5**	**1768.9**
A2-1-100-100-6	56	85.3	**831.5**	91	**84.7**	5322.5	B2-1-100-100-6	-	-	-	149	**87.0**	**2785.0**
A2-1-100-100-8	69	**85.2**	**6089.4**	-	-	-	B2-1-100-100-8	-	-	-	-	-	-
Mean	52	85.4	815.7	76	**85.2**	**762.4**	Mean	51	**86.0**	1575.7	88	**86.0**	**1051.6**
C1-1-25-75-4	48	75.1	14.9	69	**74.9**	**11.7**	D1-1-25-75-4	23	84.2	**4.7**	59	**82.5**	8.3
C1-1-25-75-5	43	76.3	20.2	75	**76.1**	**20.1**	D1-1-25-75-5	28	83.7	9.3	47	**83.3**	**8.3**
C1-1-25-75-6	38	77.0	27.8	66	**76.8**	**26.4**	D1-1-25-75-6	31	**83.8**	17.0	43	**83.8**	**15.5**
C1-1-25-75-8	32	78.3	**20.4**	43	**78.0**	39.3	D1-1-25-75-8	30	84.5	**14.9**	33	**84.4**	46.9
C1-1-50-50-4	54	84.3	72.4	81	**84.0**	**38.5**	D1-1-50-50-4	57	83.9	166.5	94	**83.6**	**40.4**
C1-1-50-50-5	49	85.2	155.7	81	**84.9**	**65.7**	D1-1-50-50-5	53	84.6	425.6	93	**84.4**	**81.7**
C1-1-50-50-6	57	85.6	363.0	88	**85.4**	**144.6**	D1-1-50-50-6	59	85.2	862.6	85	**85.0**	**191.8**
C1-1-50-50-8	58	**86.1**	1883.5	63	**86.1**	**649.5**	D1-1-50-50-8	55	85.8	2104.0	80	**85.6**	**1546.9**
Mean	47	81	319.7	71	**80.8**	**124.5**	Mean	42	84.5	450.6	67	**84.1**	**242.5**

5. Conclusion

In this paper, we proposed an exact method to solve the *Multi-vehicle Covering Tour Problem*. It is based on a set partitioning formulation that uses the ng-route relaxation for the pricing problem, and includes a stabilization procedure for dealing with degeneracy. We describe the specificities of column generation for the MCTP during the extension of a label, the dominance rule and especially the computation of tight completion bounds. The efficiency of our method is proven against the state-of-the-art method for the MCTP. Some instances have now been closed thanks to our algorithm. Moreover, the single-objective method has been embedded in a well-known bi-objective technique, the ϵ-constraint method, to solve the *Bi-Objective Multi-vehicle Covering Tour Problem* which aims to minimize both the total distance travelled and the maximum coverage distance allowed for the vehicles. The size of the final Pareto front for each instance shows that the bi-objective problem presents a compromise between the maximal covering tour coefficient and the final cost of the routes.

On the one hand, the next step of this work will be to improve the mono-objective method by adding some cuts to be able to close all instances and to deal with larger instances. On the other hand, the

bi-objective method could be improved by sharing information about optimal routes between the sub-problems solved for different ϵ values.

Acknowledgment

This research benefited from the support of ANR Project ANR-15-CE22-0012 Pi-Comodalité. Thanks are also due to the referees for their valuable comments.

References

C. Artigues, N. Jozefowiez, and B. M. Sarpong. Column generation algorithms for bi-objective combinatorial optimization problems with a min–max objective. *EURO Journal on Computational Optimization*, pages 1–26, 2018.

R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for the covering tour problem. In *Metaheuristic optimization via memory and evolution*, pages 59–91. Springer, 2005.

R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

J. E. Beasley. Route firstcluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

J. R. Current and D. A. Schilling. The covering salesman problem. *Transportation Science*, 23(3):208–213, 1989.

J. R. Current and D. A. Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114–126, 1994.

M. Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5): 977–978, 1994.

O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3): 229–237, 1999.

D. A. Flores-Garza, M. A. Salazar-Aguilar, S. U. Ngueveu, and G. Laporte. The multi-vehicle cumulative covering tour problem. *Annals of Operations Research*, 258(2):761–780, 2017.

M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45(4):568–576, 1997.

B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

M. T. Godinho, L. Gouveia, and T. L. Magnanti. Combined route capacity and route length models for unit demand vehicle routing problems. *Discrete Optimization*, 5(2):350–372, 2008.

M. H. Ha, N. Bostel, A. Langevin, and L.-M. Rousseau. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, 226(2):211–220, 2013.

M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42, 2000.

Y. Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics*, 1(3):296–297, 1971.

M. J. Hodgson, G. Laporte, and F. Semet. A covering tour model for planning mobile health care facilities in suhumdistrict, ghama. *Journal of Regional Science*, 38(4):621–638, 1998.

N. Jozefowiez. A branch-and-price algorithm for the multivehicle covering tour problem. *Networks*, 64(3):160–168, 2014.

N. Jozefowiez, F. Semet, and E.-G. Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, 34(7): 1929–1942, 2007.

M. Kammoun, H. Derbel, M. Ratli, and B. Jarboui. An integration of mixed vnd and vns: the case of the multivehicle covering tour problem. *International Transactions in Operational Research*, 24(3):663–679, 2017.

M. Labbé and G. Laporte. Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26:21–35, 1986.

R. Lopes, V. A. Souza, and A. S. da Cunha. A branch-and-price algorithm for the multi-vehicle covering tour problem. *Electronic Notes in Discrete Mathematics*, 44:61–66, 2013.

K. Murakami. A column generation approach for the multi-vehicle covering tour problem. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1063–1068. IEEE, 2014.

K. Murakami. Iterative column generation algorithm for generalized multi-vehicle covering tour problem. *Asia-Pacific Journal of Operational Research*, 35(04):1850021, 2018.

Z. Naji-Azimi, J. Renaud, A. Ruiz, and M. Salari. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *European Journal of Operational Research*, 222(3):596–605, 2012.

P. C. Nolz, K. F. Doerner, W. J. Gutjahr, and R. F. Hartl. A bi-objective metaheuristic for disaster relief operation planning. In *Advances in multi-objective nature inspired computing*, pages 167–187. Springer, 2010.

W. A. d. Oliveira, A. C. Moretti, and E. F. Reis. Multi-vehicle covering tour problem: building routes for urban patrolling. *Pesquisa Operacional*, 35(3):617–644, 2015.

T. A. Pham, M. H. Hà, and X. H. Nguyen. Solving the multi-vehicle multi-covering tour problem. *Computers & Operations Research*, 88:258–278, 2017.

A. Pigatti, M. P. De Aragao, and E. Uchoa. Stabilized branch-and-cut-and-price for the generalized assignment problem. *Electronic Notes in Discrete Mathematics*, 19:389–395, 2005.

B. M. Sarpong, C. Artigues, and N. Jozefowiez. Column generation for bi-objective vehicle routing problems with a min-max objective. In *ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013*, volume 33, pages 137–149. Schloss DagstuhlLeibniz-Zentrum fuer Informatik, 2013.

P. Toth and A. Tramontani. An integer linear programming local search for capacitated vehicle routing problems. In *The vehicle routing problem: Latest advances and new challenges*, pages 275–295. Springer, 2008.

F. Tricoire, A. Graf, and W. J. Gutjahr. The bi-objective stochastic covering tour problem. *Computers & Operations Research*, 39(7):1582–1592, 2012.