An ϵ -constraint column generation-and-enumeration algorithm for Bi-Objective Vehicle Routing Problems

Estèle Glize, Nicolas Jozefowiez, Sandra Ulrich Ngueveu

An ϵ-constraint column generation-and-enumeration algorithm for bi-objective vehicle routing problems

Estèle Glize[a], Nicolas Jozefowiez[b], Sandra Ulrich Ngueveu[a]

[a] *CNRS, LAAS, INSA, Toulouse INP, Toulouse, France*
[b] *LCOMS, Université de Lorraine, Metz, France*

Abstract

A way to solve bi-objective problems is to use an effective single objective algorithm embedded inside an ϵ-constraint approach. In this paper, we are interested in any Bi-Objective Vehicle Routing Problem such that if one objective is constrained, the resulting single objective optimization problem can be solved by a state-of-the-art column generation-based method. We propose mechanisms and techniques to significantly speed up the resulting algorithm. Computational experiments are conducted on two problems: the Bi-Objective Vehicle Routing Problem With Time Windows and the Bi-Objective Team Orienteering Problem with Time Windows. To the best of our knowledge, this paper presents the first exact method proposed for the first problem and the second exact method proposed for the second problem. On the first problem we demonstrate the effectiveness of the proposed mechanisms. On the second problem, our algorithm is competitive with algorithms from the literature.

Keywords: Combinatorial optimization, Bi-objective optimization, Vehicle Routing Problems, Column generation

1. Introduction

The purpose of this paper is to devise a method for *Bi-Objective Vehicle Routing Problems* (BOVRP) using one of the best column generation-based algorithm Baldacci et al. 2011 for *Vehicle Routing Problems* (VRP) in an ϵ-constraint approach Chankong and Haimes 1983.

Dantzig and Ramser (1959) introduced the VRP to model the distribution of gasoline from a bulk terminal to service stations by a fleet of vehicles. The VRP and many variants Toth and Vigo 2014 have been proposed and applied since then. The state-of-the-art reviews by Jozefowiez et al. (2008), Labadie and Prodhon (2014) and Vega-Mejía et al. (2017) point out that the definition of several objectives leads to more realistic and rich problems. The problems defined that way are classified as *Multi-Objective Vehicle Routing Problems* (MOVRP).

A MOVRP can be studied in the context of the *Multi-Objective Combinatorial Optimization* (MOCO) Ehrgott 2005. As the problems considered in this paper are BOVRP, we introduce here the main concepts of *Bi-Objective Combinatorial Optimization* (BOCO). The definitions are illustrated in Figure 1. A BOCO problem can be defined by an objective vector c composed of two objective functions c^1 and c^2 to minimize. It searches for specific solutions, contained in the feasible solution set $\mathcal{X}$, which are not Pareto dominated by another feasible solution. Such a solution is called an efficient (or Pareto optimal) solution. The Pareto dominance between two solutions is defined as follows:

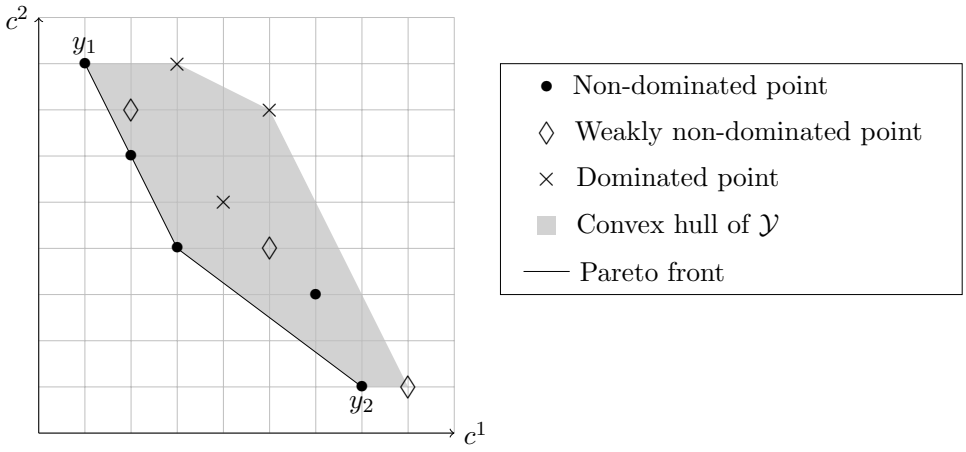


Figure 1: Specification of points in the objective space of a bi-objective minimization problem. y_1 is the c^1-extreme point and y_2 is the c^2-extreme point.

Definition 1. *Let a and b be two solutions in $\mathcal{X}$. The solution a strongly dominates ($<$) b if and only if $\forall i \in \{1,2\},\ c^i(a) < c^i(b)$. Moreover, a (weakly) dominates ($\leq$) b if and only if $\forall i \in \{1,2\},\ c^i(a) \leq c^i(b)$ and $\exists i \in \{1,2\},\ c^i(a) < c^i(b)$.*

The set $\mathcal{Y} = c(\mathcal{X}) = (c^1(\mathcal{X}), c^2(\mathcal{X}))$ is the image of the feasible solutions in the objective space. Each feasible solution $x \in \mathcal{X}$ corresponds to a single point $y = c(x)$ in the objective space, whereas a point in $\mathcal{Y}$ may be associated with several feasible solutions. The notion of dominance between solutions can be extended to their image in the objective space. However, the usual vocabulary refers to non-dominated points instead of efficient points. The non-dominated points form the non-dominated set $\mathcal{Y}_N$. Among the non-dominated points, we distinguish two points referred to as the extreme points associated with extreme feasible solutions. More precisely, the point that optimizes lexicographically c_1 then c_2 (respectively, c_2 then c_1) is called the c_1-extreme point (respectively, c_2-extreme point). Furthermore, points located on the boundary of the convex hull of $\mathcal{Y}$ are called supported non-dominated points.

1.1. Literature on Bi-Objective Combinatorial Optimization

A popular class of methods to solve MOCO problems is composed of criterion space search methods which work on the objective space. These methods usually solve a sequence of mono-objective problems and rely on effective single objective algorithms Boland et al. 2015a. According to Ehrgott and Gandibleux (2000), a well-known example is the *two-phase method* of Ulungu and Teghem (1995). It relies on the *weighted sum method* Aneja and Nair 1979 to compute the supported non-dominated points. Then, the points are sorted according to one objective, and it uses dedicated methods, such as branch-and-bound (B&B) or dynamic programming algorithms, to find non-dominated points in the area defined by two consecutive supported non-dominated points. However, the most popular algorithms are those using the ϵ-constraint method Haimes et al. 1971 as they are easy to implement while being very effective. All the objectives except one are bounded with constants, the ϵ values, and the resulting single objective optimization problem is

solved. The non-dominated set can be computed by changing the bounds on the objectives. Sáez-Aguado and Trandafir (2018) have done a review of the enhancements of the basic ϵ-constraint method in the literature. Recently, Boland et al. (2015a) have conceived the *balanced box method* to efficiently solve bi-objective mixed-integer programming problem. They consecutively explore rectangle search area in the objective space by modifying the objective to optimize. Similar ideas have been exploited in the *triangle splitting method* of Boland et al. (2015b). The other class of methods to solve MOCO problems is composed of the methods dividing the solution space such as branch-and-bound methods (B&B). Many improvements have been proposed in the literature as shown in the complete review of Przybylski and Gandibleux 2017.

1.2. Literature on Multi-Objective Vehicle Routing Problems

Many MOVRPs have been studied in the literature, and most of them aim to minimize the sum of the cost of the arcs used by a vehicle, like the mono-objective problem. Adding other objectives allows to provide some ecological Molina et al. 2014 Demir et al. 2014, workload balance Halvorsen-Weare and Savelsbergh 2016, Matl et al. 2019 or security Bula et al. 2019 concerns. However, few exact methods have been applied to MOVRPs, and the ϵ-constraint method has been mostly used. Some authors used the ϵ-constraint method combined with an integer linear programming solver to solve the resulting mono-objective compact formulation on a *Capacitated-Location Routing Problem* Toro et al. 2017, a *Multi-Objective Generalized Consistent VRP* Kovacs et al. 2015, a *Bi-Objective Hazardous Waste Location-Routing Problem* Yu and Solvang 2016 and an *Inventory-Routing Problem* Arab et al. 2018.

The ϵ-constraint method has also been used with dedicated methods to solve the resulting mono-objective problem. Reiter and Gutjahr (2012) design a branch-and-cut method (B&C) to solve a BOVRP minimizing the total cost of the routes and the difference between the longest and the shortest route. The instances they use have between 16 and 57 nodes and 2 to 9 available vehicles. They solve 10 instances out of 57 in 8 hours. Halvorsen-Weare and Savelsbergh (2016) use a dynamic programming algorithm to solve a BOVRP minimizing the total cost and different balance routing objective. Their instances are up to 12 nodes. Tricoire (2012) solves a *Bi-Objective Stochastic Covering Tour Problem* with a B&C. The instances with 30 nodes have been solved in a time limit of 3 days. The number of non-dominated points varies between 6 and 112.

Finally, Parragh and Tricoire (2019) introduce a new B&B to solve generic *Bi-Objective Integer Programming* problems. They apply their method to solve a *Bi-Objective Team-Orienteering Problem* minimizing the total cost and maximizing the profit collected by visiting a node (see Section 4.1). Basic column generation technique has been added to solve the linear relaxation of the mono-objective formulations. The instances have between 15 and 35 nodes, and the B&B solves 69 instances out of 80 in 2 hours. In their paper, they also compare the following criterion space search methods: the ϵ-constraint, a bi-directional ϵ-constraint method and the *balanced box method* both described in Boland et al. (2015a). They show that the ϵ-constraint method is more effective on two different *Uncapacitated Bi-Objective Facility Location* problems than its bi-directional counterpart and the *balanced box method*. They also show that the B&B was the most effective method for all considered problems.

In this paper, one of the most effective algorithms for vehicle routing problems, the column generation and enumeration algorithm of Baldacci et al. (2011), is embedded in an ϵ-constraint method. The main contribution of this paper is the design of five mechanisms to significantly speed-up the method. These mechanisms exploit properties of the column generation algorithms

and the decomposition in the context of the ϵ-constraint approach. Unlike classical criterion space search methods, the mechanisms allow to get information from one mono-objective problem to another. Experiments show that the final algorithm is the state-of-the-art algorithm for a problem defined by Parragh and Tricoire (2019), the *Bi-Objective Team Orienteering Problem with Time Windows*. The algorithm can also serve as a benchmark for future works on *Bi-Objective Vehicle Routing Problems with Time Windows*.

The remainder of the paper is organized as follows. The class of studied BOVRP is defined in Section 2. This section also describes the basic algorithm used to solve these problems. Additional properties and performance improvement techniques are proposed in Section 3. Section 4 presents the computational results. Conclusions are provided in Section 5.

2. An ϵ-Constraint Method for Bi-Objective Vehicle Routing Problems

In 2.1, the characteristics of the BOVRPs that can be solved by our algorithm are described. A general bi-objective mathematical model is formulated in Section 2.2. As the algorithm is based on the ϵ-constraint method, the model for a fixed value of ϵ is given in Section 2.3. This mono-objective formulation is solved by the algorithm of Baldacci et al. (2011), which is outlined in Section 2.5. The basic ϵ-constraint method is described in Section 2.6.

2.1. Bi-Objective Vehicle Routing Problems

We consider BOVRP with two positive costs per edge. Therefore, a route has two costs. The first route cost (respectively, the second route cost) is the sum of the first costs (respectively, the second costs) on the edges of the route. This is required to apply the new mechanisms. Therefore, the mechanisms cannot be used with some routing problems such as the bi-objective covering tour problem Jozefowiez et al. 2007, Glize et al. 2020 even if the algortihm of Baldacci et al. can be applied to them. Operational constraints such as the maximum number of routes, time windows, vehicle capacities can be considered.

Formally, the problem is defined over a graph $G = (V, E)$. The graph may or may not be oriented. The vertex set $V = \{0, \ldots, n\}$ contains a depot 0 and n customers i, $1 \leq i \leq n$. Let $V' = V \setminus \{0\}$ be the customer set. This set can be partitioned into two subsets: i) V'_1 the set of customers that must be visited; ii) V'_2 the set of customers that may not be visited. Two integer costs c^1_{ij} and c^2_{ij} are associated with each edge $(i, j) \in E$. Such integrality condition on the costs can be relaxed , except if one of the mechanisms introduced in Section 3 is used. However, this condition facilitates the design of the algorithm. The customers are visited by a set of K vehicles located at the depot. A path p, performed by one vehicle, starts at the depot and passes through a set of edges E_p. It has two costs $c^1_p = \sum_{(i,j)\in E_p} c^1_{ij}$ and $c^2_p = \sum_{(i,j)\in E_p} c^2_{ij}$. A route is a path which ends at the depot. The goal is to find a set of routes that visits all required customers, respects the operational constraints and minimizes the total costs.

2.2. Bi-Objective Mathematical Model

The problems can be modeled by a set partitioning formulation. Let R be the set of all feasible routes. Let a_{ir} be equal to 1 if route $r \in R$ visits the node $i \in V'_1$, -1 if r visits $i \in V'_2$ and 0 otherwise. For each $i \in V'$, we define a constant d_i equal to 1 if $i \in V'_1$ and -1 otherwise. A binary variable x_r is equal to 1 if and only if the route $r \in R$ is selected. The model is:

$$
\begin{cases}
\text{minimize} \left(\sum_{r \in R} c_r^1 x_r, \sum_{r \in R} c_r^2 x_r \right) & & (1.1) \\
s.t. \sum_{r \in R} a_{ir} x_r \geq d_i & i \in V' & (1.2) \\
\sum_{r \in R} x_r \leq |K| & & (1.3) \\
x_r \in \{0,1\} & r \in R & (1.4)
\end{cases} \tag{1}
$$

The objective vector (1.1) minimizes the two costs. Constraints (1.2) ensure that a vertex is visited once if it belongs to V_1' or visited at most once if it belongs to V_2'. Constraint (1.3) limits the number of vehicles available. Finally, Constraints (1.4) define the variable domains.

2.3. The ϵ-Constraint Mathematical Model

Model (1) is transformed into Model (2) by the introduction of a bound ϵ on the second objective.

$$
\begin{cases}
\text{minimize} \sum_{r \in R} c_r^1 x_r & & (2.1) & \\
s.t. \sum_{r \in R} a_{ir} x_r \geq d_i & i \in V' & (2.2) & \longleftrightarrow \lambda_i^\epsilon \\
\sum_{r \in R} x_r \leq |K| & & (2.3) & \longleftrightarrow \lambda_0^\epsilon \\
\sum_{r \in R} c_r^2 x_r \leq \epsilon & & (2.4) & \longleftrightarrow \alpha^\epsilon \\
x_r \in \{0,1\} & r \in R & (2.5) &
\end{cases} \tag{2}
$$

Objective (2.1) minimizes the first objective and Constraint (2.4) bounds the value of the second objective. The other constraints are as in Model (1). The dual variables of each constraint are reported on the right-hand side of the model.

2.4. Notations

Model (1) is the Master Problem (MP), and its linear relaxation is the Linear Master Problem (LMP). Model (2) is the ϵ-Master Problem (MP(ϵ)), and its linear relaxation is the ϵ-Linear Master Problem (LMP(ϵ)). The dual of LMP(ϵ) is denoted DLP(ϵ). These models restricted on a subset of routes $\overline{R} \subseteq R$ will be denoted RMP(ϵ,$\overline{R}$), RLMP(ϵ,$\overline{R}$) and RDLP(ϵ,$\overline{R}$). The pair $\Lambda^\epsilon = (\lambda^\epsilon, \alpha^\epsilon)$ represents the dual variables associated with Constraints (2.2), Constraint (2.3) and Constraint (2.4) for a fixed value of ϵ, respectively. According to the context, Λ^ϵ can also represent the dual values obtained by the solution of RLMP(ϵ,$\overline{R}$). To lighten the notation, if x (respectively, y) is a feasible solution (respectively, a point in the objective space), the objective values (respectively, the point coordinates) will be c_x^1 and c_x^2 (respectively, c_y^1 and c_y^2).

2.5. Solution of MP(ϵ)

Model (2) has an exponential number of variables x_r. However, it can be solved by the column generation and enumeration algorithm introduced by Baldacci et al. (2008) and later improved by Baldacci et al. (2011). The algorithm is composed of the following four steps, and it will be denoted by `GENROUTE` in this paper.

2.5.1. Step 1: Computation of the Lower Bound

The optimum of LMP(ϵ) is a lower bound LB of MP(ϵ) and is obtained by column generation. Usually, the pricing problem of the column generation for VRP is an *Elementary Shortest Path problem with Resource Constraints*, and it is solved with a labelling algorithm which produces elementary routes with negative reduced cost. As in Baldacci et al. (2011), the elementarity constraint on the routes is relaxed, meaning that a route can visit a customer more than once. The relaxation is done through the definition of ng-routes. An ng-route allows several visits to the same customer as long as two visits are not close in the sequence. A parameter, called the size of the ng-sets, controls when it is possible to revisit a customer. Additional details on this relaxation and how to implement it can be found in Baldacci et al. (2011).

This step is enclosed in the function *compute_lowerbound*. The input of the function is the bound on the second objective ϵ. The output is a lower bound LB on the first objective, the set of ng-routes $\widetilde{R}$ in the restricted master problem at the end of the algorithm and the dual values Λ^ϵ associated with the optimum of RLMP(ϵ, $\widetilde{R}$).

2.5.2. Step 2: Computation of an Upper Bound

The second step aims to find an upper bound UB which is a point in the objective space associated with a feasible solution of MP(ϵ). To do so, the method used dedicated heuristics for each problem which are explained in Sections 4.2 and 4.3. In the next algorithms, the function *compute_upperbound* takes ϵ in argument and returns a point UB in the objective space along with an associated solution.

2.5.3. Step 3: Route Enumeration

Given the dual values Λ^ϵ obtained at the end of the first step, we search for the routes with a reduced cost smaller or equal to the value $\gamma = UB - LB$. This value is called the gap. Baldacci et al. (2008) introduced this third step called column enumeration. The enumeration is done by means of a monodirectional or a bidirectional labelling algorithm. In a labelling algorithm, a label represents a path by a subset of features called resources (like its load, its reduced cost, its last visited customer...). It is progressively extended to another label by visiting other customers. The size of the set of labels is managed by dominance rules and completion bounds. The dominance rule allows to discard labels that will not lead to a better solution than other labels. For a given label, the completion bound represents the lower bound on reduced cost of paths ending at the depot and that can complement the current label to form a feasible route. Therefore, the completion bounds allow to remove labels whose reduced cost would be greater than the gap.

In the next algorithms, this step corresponds to the function *route_enumeration*. The input of the function is the gap γ and the dual values Λ^ϵ. The output is a set of columns $\overline{R}$ with a reduced cost less than or equal to the gap.

2.5.4. Step 4: Computation of an Optimal Solution

The optimum and an optimal solution of MP(ϵ) can be obtained by solving RMP(ϵ, $\overline{R}$). The integer problem can be solved with a black box MILP solver. In the algorithms, this corresponds to a call to the function *solve*. The inputs are ϵ and $\overline{R}$, and the output is an optimal solution for MP(ϵ).

Compute the extreme points $S1$ and $S2$;
$\mathcal{Y}_N = \{(c^1_{S1}, c^2_{S1}), (c^1_{S2}, c^2_{S2})\}$;
$i = 1$; $\epsilon_i = c^2_{S1} - 1$; $UB_{i-1} = S1$;
while $c^1_{UB_{i-1}} < c^1_{S2}$ **do**
  /* Step 1 (Section 2.5.1) */
  $LB_i, \Lambda^{\epsilon_i} \leftarrow$ *compute_lowerbound*(ϵ_i) ;
  /* Step 2 (Section 2.5.2) */
  $UB_i \leftarrow$ *compute_upperbound*(ϵ_i) ;
  /* Step 3 (Section 2.5.3) */
  $\gamma_i = \min(c^1_{S2}, c^1_{UB_i}) - LB_i$;
  $\overline{R}_i \leftarrow$ *route_enumeration*$(\gamma_i, \Lambda^{\epsilon_i})$;
  /* Step 4 (Section 2.5.4) */
  $O_i \leftarrow$ *solve*$(\epsilon_i, \overline{R}_i)$;
  /* Update the set of non-dominated points and the next ϵ value */
  $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{(c^1_{O_i}, c^2_{O_i})\}$;
  $\epsilon_{i+1} = c^2_{O_i} - 1$;
  $i \leftarrow i + 1$;
end
$\overline{R} \leftarrow \overline{R}_{i-1}$;
while $\epsilon_i > c^2_{S2}$ **do**
  /* Step 4 (Section 2.5.4) */
  $O_i \leftarrow$ *solve*$(\epsilon_i, \overline{R})$) ;
  /* Update the set of non-dominated points and the next ϵ value */
  $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{(c^1_{O_i}, c^2_{O_i})\}$;
  $\epsilon_{i+1} = c^2_{O_i} - 1$;
  $i \leftarrow i + 1$;
end
Return $\mathcal{Y}_N$;

Algorithm 1: Stepwise ϵ-constraint method.

2.6. Stepwise ϵ-Constraint Method

GENROUTE can easily be embedded in an ϵ-constraint approach. The resulting algorithm is called *Stepwise ϵ-constraint method* (SeM) given in Algorithm 1. The output $\mathcal{Y}_N$ is the minimal complete set Hansen 1980 for Model (1). That is the set composed of all non-dominated points in the objective space and at least one feasible solution for each point.

The algorithm starts by computing the two extreme solutions $S1$ and $S2$. This can be done using GENROUTE with the objectives $\min c^1 + \alpha_1 c^2$ and $\min c^2 + \alpha_2 c^1$ with α_1 and α_2 small enough constants. These lexicographic objectives ensure that the extreme points returned are not weakly dominated. As c^1 is an integer cost, it can be deduced that $\alpha_1 \in]0; \frac{1}{c^2(S1)}]$. Let ω be an upper bound on the cost c^2 of any non-dominated point - for instance, $\omega = \sum_{i \in V'} (c^2_{0i} + c^2_{i0})$. Thus, α_1 can be fixed to $\frac{1}{\omega}$. The same goes for α_2. In practice, we use $\alpha_1 = \alpha_2 = 10^{-5}$ which is valid for all the instances.

After computing $S1$ and $S2$, an iterative process begins with $i = 1$ and $O_0 = S1$. Also, ϵ_1 is

fixed such that S_1 is excluded. The i^{th} iteration is composed of the following actions. First, an optimal solution O_i of MP(ϵ_i) is found by `GENROUTE` (steps 1 to 4). Then, the point $(c^1_{O_i}, c^2_{O_i})$ is added to $\mathcal{Y}_N$, ϵ_{i+1} is fixed to exclude O_i, and i is incremented. The algorithm stops when $c^2_{O_i} \leq c^2_{S2}$.

Algorithm 1 can be sped up as follows. First, if $c^1_{UB_i}$ exceeds c^1_{S2}, the gap γ_i in step 3 can be tightened to $c^1_{S2} - LB_i$. Indeed, no non-dominated point can have a first objective value strictly greater than the one of S_2. Moreover, at the end of the first loop of Algorithm 1, the set $\overline{R}_{i-1}$ contains all routes with a reduced cost less than or equal to $c^1_{S2} - LB_{i-1}$. So, the routes in $\overline{R}_{i-1}$ are sufficient to compute the remaining non-dominated points without the generation of new routes. This assertion is proved in Appendix A. Therefore, the remaining non-dominated points can be generated by solving the restricted master problem RMP(ϵ,$\overline{R}_{i-1}$) until the value of ϵ is less than or equal to c^2_{S2}. As a final remark, weakly dominated solutions can be generated by the algorithm and are removed from $\mathcal{Y}_N$.

3. The ϵ-Constraint Column Generation-and-Enumeration Algorithm

We propose mechanisms to improve the efficiency of SeM. The mechanisms are presented in Section 3.1-3.5. The inclusion of these mechanisms in SeM leads to an algorithm, the ϵ-constraint Column Generation-and-Enumeration Algorithm (eCGEA), described in Section 3.6.

3.1. T_{gap}: Reduction of the Gap

At each iteration of SeM, a gap γ is computed between the lower bound and an upper bound. This gap bounds the value of the reduced cost of the routes generated during the route enumeration phase. Therefore, tightening the gap should improve the efficiency of the enumeration. To reduce the gap during iteration i, we set γ_i to $c^1_{UB_i} - LB_i - 1$. Indeed, the sum of the reduced costs of the routes of a feasible solution O is less than the difference between c^1_O and the lower bound. Decreasing the gap by one is possible because if no feasible solution exists in the resulting restricted master problem, the solution corresponding to the upper bound is optimal. Note that this reduction is possible because the cost matrix c^1 has integer coefficients. For instance, let the lower bound be 2.5 and the upper bound value on the first objective be 5. With this technique, the gap γ will be 1.5. Therefore, all feasible solutions having a cost c^1 less than or equal to 4 can be found in the resulting restricted master problem. One could want to compute the gap between the lower bound rounded up (3) and the upper bound minus one. However, a gap of 1 generates solutions with a cost c^1 less than 3.5, and a feasible solution of cost 4 could be missed. If this mechanism is used, the integrality condition of the cost c^1 cannot be relaxed.

3.2. T_{lb}: Avoidance of Lower Bound Computation

A possibility to improve the computational times is to limit the number of calls to the column generation algorithm that computes a lower bound, i.e. the first step of `GENROUTE`. Let LB_i be the lower bound, $\Lambda^{\epsilon_i} = (\lambda^{\epsilon_i}, \alpha^{\epsilon_i})$ be the dual values and $\widetilde{R}_i$ be the set of routes returned by the function *compute_lowerbound(ϵ_i)* during iteration i of SeM. A sensitivity analysis can provide an interval $[\Delta^i_a; \Delta^i_b]$ on the value of ϵ in which the optimal basis of RLMP(ϵ_i,$\widetilde{R}_i$) and α^{ϵ_i} do not change. Therefore, the set $\widetilde{R}_i$ can be used to compute the lower bound for a value ϵ_j in $[\Delta^i_a; \epsilon_i - 1]$. The optimum of LMP($\epsilon_j$) is directly given by $LB_j = LB_i - (\epsilon_i - \epsilon_j)\alpha^{\epsilon_i}$. Note that the dual values Λ^{ϵ_i} are not necessarily an ideal choice for another ϵ_j value, even if ϵ_j in $[\Delta^i_a; \Delta^i_b]$. Therefore, the impact of this mechanism on the global efficiency of the algorithm may vary from one problem to another.

The function *compute_lowerbound*, introduced in Section 2.5.1, is modified to check the condition of the sensitivity analysis and avoid the generation of a new restricted master problem. The dual values $\Lambda^{\epsilon_{i-1}}$ and the set of routes $\widetilde{R}_{i-1}$ used during the previous iteration are additional inputs to the method.

3.3. T_{valid}: Avoidance of Route Enumeration

Another costly part of SeM is the route enumeration done at each iteration. The following proposition gives the conditions for which the upper bound computed at iteration i is a non-dominated point.

Proposition 1. *If $\lambda_0^{\epsilon_{i-1}} = \lambda_0^{\epsilon_i}$, $\alpha^{\epsilon_{i-1}} = \alpha^{\epsilon_i}$ and $\gamma_i \leq \gamma_{i-1}$, then the optimal solution of RMP(ϵ_i,$\overline{R}_{i-1}$) is an optimal solution of MP(ϵ_i). It is also an efficient solution.*

To prove this proposition, we need to prove the following theorem. To lighten the notations, we pose $i = 2$. The reduced cost of a route r, computed with respect to the dual variables Λ^{ϵ_1}, is noted $\overline{c_r}(\Lambda^{\epsilon_1})$. Theorem 1 proves that the sum of the reduced costs of the routes of any feasible solution UB_2 with respect to Λ^{ϵ_1} is the same as this sum expressed according to Λ^{ϵ_2}, if $\lambda_0^{\epsilon_1} = \lambda_0^{\epsilon_2}$ and $\alpha^{\epsilon_1} = \alpha^{\epsilon_2}$. Therefore, the routes of UB_2 have the sum of their positive reduced costs with respect to Λ^{ϵ_1} and Λ^{ϵ_2} less than or equal to γ_2. If $\gamma_2 \leq \gamma_1$, the routes of UB_2 have already been generated in the route enumeration between LB_1 and UB_1, so there is no need to launch the route enumeration procedure.

Theorem 1. *Let UB_2 be a feasible solution of MP(ϵ_1) and Λ^{ϵ_1} (respectively Λ^{ϵ_2}) an optimal solution of DLP(ϵ_1) (respectively DLP(ϵ_2)) with $\epsilon_1 > \epsilon_2$. If $\lambda_0^{\epsilon_1} = \lambda_0^{\epsilon_2}$ and $\alpha^{\epsilon_1} = \alpha^{\epsilon_2}$, then the sum of the reduced costs of UB_2 with respect to Λ^{ϵ_2} can be computed as:*

$$\textstyle\sum_{r\in R} \overline{c_r}(\Lambda^{\epsilon_2})\ x_r^{UB_2} = \sum_{r\in R}(c_r^1 - \sum_{i\in V} a_{ir}\lambda_i^{\epsilon_1} - c_r^2\alpha^{\epsilon_1})x_r^{UB_2}$$

Proof.

$$\begin{aligned}
\sum_{r\in R} \overline{c_r}(\Lambda^{\epsilon_2})\ x_r^{UB_2} &= \sum_{r\in R}(c_r^1 - \sum_{i\in V} a_{ir}\lambda_i^{\epsilon_2} - c_r^2\alpha^{\epsilon_2})x_r^{UB_2} \\
&= \sum_{r\in R} c_r^1 x_r^{UB_2} - \sum_{i\in V}(\sum_{r\in R} a_{ir}x_r^{UB_2})\lambda_i^{\epsilon_2} - \sum_{r\in R} c_r^2\alpha^{\epsilon_2}x_r^{UB_2} \\
&= \sum_{r\in R} c_r^1 x_r^{UB_2} - \sum_{i\in V'} \lambda_i^{\epsilon_2} - \sum_{r\in R} x_r^{UB_2}\lambda_0^{\epsilon_2} - \sum_{r\in R} c_r^2\alpha^{\epsilon_2}x_r^{UB_2} \\
&= \sum_{r\in R} c_r^1 x_r^{UB_2} - \sum_{i\in V'} \lambda_i^{\epsilon_1} - \sum_{r\in R} x_r^{UB_2}\lambda_0^{\epsilon_1} - \sum_{r\in R} c_r^2\alpha^{\epsilon_1}x_r^{UB_2} \quad \textit{[by Proposition 2]} \\
&= \sum_{r\in R}(c_r^1 - \sum_{i\in V} a_{ir}\lambda_i^{\epsilon_1} - c_r^2\alpha^{\epsilon_1})x_r^{UB_2}
\end{aligned}$$

□

Theorem 1 uses Proposition 2 which gives the conditions for which the sum of the dual variables of Λ^{ϵ_1} and Λ^{ϵ_2} are equal.

Proposition 2. *Let LB_1 (respectively LB_2) be the lower bound of MP(ϵ_1) (respectively MP(ϵ_2)) and Λ^{ϵ_1} (respectively Λ^{ϵ_2}) an optimal solution of DLP(ϵ_1) (respectively DLP(ϵ_2)) with $\epsilon_1 > \epsilon_2$. If $\lambda_0^{\epsilon_1} = \lambda_0^{\epsilon_2}$ and $\alpha^{\epsilon_1} = \alpha^{\epsilon_2}$, then $\sum_{i \in V'} \lambda_i^{\epsilon_1} = \sum_{i \in V'} \lambda_i^{\epsilon_2}$.*

Proof. According to Section 3.2, we obtain:

$$
\begin{aligned}
& c^1_{LB_1} - (c^2_{LB_1} - c^2_{LB_2})\alpha^{\epsilon_1} = c^1_{LB_2} \\
\Rightarrow & \sum_{r \in R} c^1_r x_r^{LB_1} - (\epsilon_1 - \epsilon_2)\alpha^{\epsilon_1} = \sum_{r \in R} c^1_r x_r^{LB_2} \\
\Rightarrow & \sum_{r \in R} c^1_r x_r^{LB_1} - \epsilon_1 \alpha^{\epsilon_1} = \sum_{r \in R} c^1_r x_r^{LB_2} - \epsilon_2 \alpha^{\epsilon_2} \ [\alpha^{\epsilon_2} = \alpha^{\epsilon_1}] \\
\Rightarrow & \sum_{i \in V'} \lambda_i^{\epsilon_1} + K\lambda_0^{\epsilon_1} = \sum_{i \in V'} \lambda_i^{\epsilon_2} + K\lambda_0^{\epsilon_2} \ [\textit{strong duality theorem}] \\
\Rightarrow & \sum_{i \in V'} \lambda_i^{\epsilon_1} = \sum_{i \in V'} \lambda_i^{\epsilon_2} \ [\lambda_0^{\epsilon_2} = \lambda_0^{\epsilon_1}]
\end{aligned}
$$

□

Furthermore, it is possible to improve the computational times by removing from $\overline{R}_{i-1}$ routes that cannot be part of a solution for MP(ϵ_i), i.e. the routes with a reduced cost, computed according to Λ^{ϵ_i}, greater than γ_i. This routine is enclosed in the function *column_deletion*. The inputs are the dual values Λ^{ϵ_i}, a gap γ^{ϵ_i} and a set of routes $\overline{R_{i-1}}$. The output is a new set $\overline{R_i}$ of routes from $\overline{R_{i-1}}$ whose reduced costs, computed with respect to Λ^{ϵ_i}, are less than γ^{ϵ_i}.

3.4. T_{save}: Warm Start Route Enumeration

Another improvement is to avoid starting the labelling algorithm with an empty route, but instead to warm start it using labels generated during previous enumeration phases. At iteration i, the routes and labels generated during the previous iterations can be partitioned into four sets:

- $\overline{R}_{i-1}$: feasible routes with a reduced cost inferior to the gap;
- $\overline{Lr}_{i-1}$: feasible routes dominated by another feasible route or with a reduced cost greater than the gap;
- $\overline{Lp}_{i-1}$: feasible paths dominated by another path, eliminated by a completion bound, or infeasible with respect to the cost c^2;
- $\overline{I}_{i-1}$: labels leading to infeasible routes with respect to resources other than the cost c^2 (e.g., time windows, vehicle capacities ...). This set is not saved.

The enumeration phase at iteration i can be started with labels from $\overline{R}_{i-1}$, $\overline{Lr}_{i-1}$ and $\overline{Lp}_{i-1}$. First, the reduced costs of the routes of the two sets, $\overline{Lr}_{i-1}$ and $\overline{R}_{i-1}$, are computed with respect to Λ^{ϵ_i}. Then, these routes are stored into two sets. The first set $\overline{R}_i$ contains the ones with a reduced cost less than γ_i. The remaining ones are stored in $\overline{Lr}_i$. This first routine is an extension of the function *column_deletion*, introduced in Section 3.3, with only one extra input: the set $\overline{Lr}_{i-1}$.

After that, the enumeration algorithm is started from $\overline{Lp}_{i-1}$ instead of a label representing an empty route starting at the depot. During this enumeration phase, the sets $\overline{R}_i$, $\overline{Lr}_i$ and $\overline{Lp}_i$ are

completed with the adequate routes and paths. Initially, the sets $\overline{R}_0$ and $\overline{Lr}_0$ are empty, and $\overline{Lp_0}$ is initialized with a label representing a visit to the depot. As the sets may become too large, they are reset between two iterations $i-1$ and i if $\lambda_0^{\epsilon_i} \neq \lambda_0^{\epsilon_{i-1}}$ or $\alpha^{\epsilon_i} \neq \alpha^{\epsilon_{i-1}}$. Moreover, if the route enumeration is warm started, the search is necessarily monodirectional as a bidirectional algorithm would require to store too many initialization labels. In the algorithm, the function *route_enumeration*, introduced in Section 2.5.3, is modified to accept an additional parameter that is the set of labels to initialize the search. If the search is not warm started, this parameter is the empty set. This function also returns the new sets $\overline{Lp}_i$ and $\overline{Lr}_i$. It is important to return $\overline{Lr}_i$ because the routes not used at iteration i can be useful during subsequent iterations.

3.5. T_{dual}: Multiple Reduced Cost

One of the main drawbacks of `GENROUTE` is the sensitivity of the route enumeration procedure to the gap. The algorithm may not converge because of a small increase of the gap. If this happens during SeM, the algorithm would be stuck even though the problems for subsequent ϵ values may be solved. The following strategy is used to mitigate this issue. The number of columns generated during the enumeration phase is bounded by a value MAX_COL. That way, the algorithm can generate a set of points, not necessarily non-dominated, in the objective space. To that end, the subroutine *route_enumeration* is modified to accept an additional parameter that is the maximum number of routes to generate.

However, if the enumeration stops because the bound MAX_COL is reached at iteration i, then the solution O_i of RMP(ϵ_i, $\overline{R}_i$) may not be an efficient solution. The missing non-dominated points, located in the rectangle defined by (LB_i, ϵ_i) and $P_i = (c^1_{O_i}, c^2_{O_i})$, are searched during a second enumeration phase. To perform this new enumeration, multiple dual values are considered. To that end, the following linear problems (3) are solved by column generation for a set of weights $W = \{w_k : 0 \leq k \leq |W|\}$ that are supposed to be sorted increasingly and with $w_0 = 0$ and $w_{|W|} = 1$. The set W is a parameter of eCGEA.

$$
\begin{cases}
min \sum_{r \in R} (w_k * c^1_r x_r + (1 - w_k) * c^2_r x_r) & & (3.1) & \\
s.t. \sum_{r \in R} a_{ir} x_r \geq d_i & i \in V' & (3.2) & \longleftrightarrow \lambda_i^{w_k} \\
\sum_{r \in R} x_r \leq |K| & & (3.3) & \longleftrightarrow \lambda_0^{w_k} \\
\sum_{r \in R} c^1_r x_r \leq c^1_{O_i} & & (3.4) & \longleftrightarrow \alpha_1^{w_k} \\
\sum_{r \in R} c^2_r x_r \leq \epsilon_i & & (3.5) & \longleftrightarrow \alpha_2^{w_k} \\
x_r \geq 0 & r \in R & (3.6) &
\end{cases}
\tag{3}
$$

For each weight w_k, the solution of Model (3) gives a point in the objective space LB_{w_k} as well as the dual values $\Lambda^{w_k} = (\lambda^{w_k}, \alpha^{w_k})$. A gap γ_{w_k} with respect to the point $(c^1_{O_i}, \epsilon_i)$ is computed. This is illustrated in Figure 2 for $W = \{0, 0.5, 1\}$. In the final algorithm, this step corresponds to a call to a modified *compute_lowerbound* function in which the weight w_k, the solution O_i and ϵ_i are additional inputs.

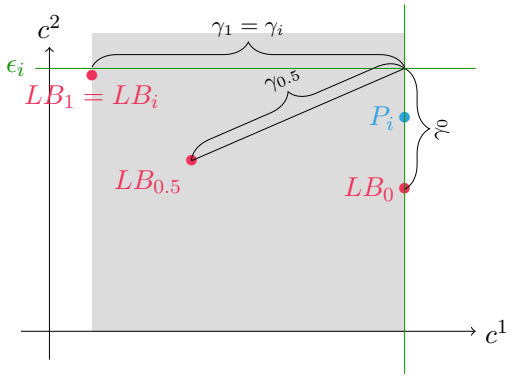


Figure 2: Lower bounds and gaps computed with T_{dual} for $W = \{0, 0.5, 1\}$ because generated columns in the gap γ_1 are more than MAX_COL

That way, a route r has $|W|$ reduced costs which are additional resources for the route enumeration algorithm. A label can be discarded if at least one of the reduced costs is greater than the gap associated to the direction. This can also be improved by using a different completion bound for each reduced cost. In the algorithm, this corresponds to a variant of the function *route_enumeration* where the inputs consist of the set of weights W, the dual values for each LB_{w_k}, the gap for each weight w_k and O_i. The enumeration returns a set of routes $\overline{R}_i$. An ϵ-constraint algorithm is performed to find the missing non-dominated points by solving RMP(ϵ,$\overline{R}_i$) for $c^2_{O_i} \leq \epsilon \leq \epsilon_i - 1$.

3.6. Complete Algorithm

The mechanisms presented above are included in SeM to improve its efficiency. The new solution method is called ϵ-constraint Column Generation-and-Enumeration Algorithm (eCGEA). Algorithms 2 and 3 give the pseudo-code. Initially, the extreme points are computed as in SeM. Then, at each iteration, ϵ is set such that the previous solution is excluded and a new solution is computed. A difference with SeM is the computation of the gap following T_{gap}. Moreover, tests are performed during the while loop to avoid some calls to *compute_lowerbound* with T_{lb} and *route_enumeration* with T_{valid}. If a call of *route_enumeration* is made, T_{save} is used to warm start the labelling algorithm. Also, if the *route_enumeration* procedure is stopped because the bound MAX_COL is reached, its index is saved in the set I. The second loop of Algorithm 2 corresponds to the T_{dual} mechanism. For each iteration where MAX_COL was reached during the enumeration, the rectangle defined by $(c^1_{O_{i-1}}, c^2_{O_{i-1}})$ and $(c^1_{O_i}, c^2_{O_i})$ is explored to search for missing non-dominated points. A lower bound for each weight in W is computed, and another enumeration is performed as explained in 3.5. Then, the restricted master problem is solved by an ϵ-constraint algorithm on the complete rectangle.

4. Computational Results

The experiments have been conducted on a Xeon E5-2695 processor with a 2.30GHz CPU in a single thread. The implementation is in C++, and the linear problems and the integer problems are solved with Gurobi 7.1. Two problems have been defined to evaluate the efficiency of the algorithm. The first problem, the Bi-Objective Vehicle Routing Problem with Time Windows (BOVRPTW), is

Compute the extreme points $S1$ and $S2$;
$\mathcal{Y}_N = \{(c^1_{S1}, c^2_{S1}), (c^1_{S2}, c^2_{S2})\}$;
$i = 0$; $\epsilon_i = c^2_{S1} - 1$;
$\overline{R}_{i-1} \leftarrow \emptyset$, $\overline{Lr}_{i-1} \leftarrow \emptyset$, $\overline{Lp}_{i-1} \leftarrow \{0\}$, $\widetilde{R}_{i-1} \leftarrow \emptyset$, $\Lambda^{\epsilon_{i-1}} \leftarrow 0$, $I \leftarrow \emptyset$;
while $\epsilon_i > c^2_{S2}$ **do**
  /* Step 1 (Section 2.5.1) with T_{lb} (Section 3.2) */
  $LB_i, \widetilde{R}_i, \Lambda^{\epsilon_i} \leftarrow compute\_lowerbound(\epsilon_i, \widetilde{R}_{i-1}, \Lambda^{\epsilon_{i-1}})$;
  /* Step 2 (Section 2.5.2) */
  $UB_i \leftarrow compute\_upperbound(\epsilon_i)$;
  /* Step 3 (Section 2.5.3) with T_{gap} (Section 3.1) */
  $\gamma_i = \min(c^1_{S2}, c^1_{UB_i}) - LB_i - 1$;
  $\overline{R}_i, \overline{Lr}_i, \overline{Lp}_i \leftarrow generate\_route(\gamma_i, \gamma_{i-1}, \Lambda^{\epsilon_{i-1}}, \Lambda^{\epsilon_i}, \overline{R_{i-1}}, \overline{Lr_{i-1}}, \overline{Lp_{i-1}})$;
  /* Step 4 (Section 2.5.4) */
  $O_i \leftarrow solve(\epsilon_i, \overline{R}_i)$;
  /* Update the set of non-dominated points and the next ϵ value */
  $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{(c^1_{O_i}, c^2_{O_i})\}$;
  $\epsilon_{i+1} = c^2_{O_i} - 1$; $i \leftarrow i + 1$;
  /* For T_{dual}, check the size of $\overline{R_i}$ (Section 3.5) */
  if $|\overline{R_i}| \geq MAX\_COL$ **then**
    $I \leftarrow I \cup \{i\}$;
  end
end
/* For T_{dual}, generate other reduced costs for non-explored area (Section 3.5) */
for $i \in I$ **do**
  /* Non-explored area = rectangle defined by $(c^1_{O_{i-1}}, c^2_{O_{i-1}})$ and $(c^1_{O_i}, c^2_{O_i})$ */
  for $w \in W$ **do**
    /* Step 1 (Section 2.5.1) */
    $LB^w, \Lambda_w \leftarrow compute\_lowerbound(w, O_i, \epsilon_i)$;
    $\gamma_w = (w * c^1_{O_i} + (1 - w) * c^2_{O_{i-1}}) - (w * c^1_{LB_w} + (1 - w) * c^2_{LB_w})$;
  end
  /* Step 3 (Section 2.5.3) with T_{dual} (Section 3.5) */
  $\overline{R} \leftarrow route\_enumeration(\gamma_0, \lambda_0, .., \gamma_{|W|}, \lambda_{|W|}, O_i)$;
  /* ϵ-constraint in the complete non-explored area */
  while $\epsilon > c^2_{O_i}$ **do**
    $P \leftarrow solve(\epsilon, \overline{R})$;
    /* Update the set of non-dominated points and the next ϵ value */
    $\mathcal{Y}_N \leftarrow \mathcal{Y}_N \cup \{(c^1_P, c^2_P)\}$;
    $\epsilon = c^2_P - 1$;
  end
end
Return $\mathcal{Y}_N$;

Algorithm 2: ϵ-constraint Column Generation-and-Enumeration Algorithm.

/* Check if possible not to do the step 3 or to warm start the step 3 */
if $\lambda_0^{\epsilon_{i-1}} \neq \lambda_0^{\epsilon_i}$ *or* $\alpha^{\epsilon_{i-1}} \neq \alpha^{\epsilon_i}$ *or* $|\overline{R}_{i-1}| \geq MAX\_COL$ **then**
  /* Step 3 (Section 2.5.3) starting the labelling algorithm with an empty route and limiting the size of $\overline{R}_i$ to MAX_COL (Section 3.5) */
  $\overline{R}_i, \overline{Lr}_i, \overline{Lp}_i \leftarrow route\_enumeration(\gamma_i, \Lambda^{\epsilon_i}, MAX\_COL)$;
else
  /* Remove non interesting routes at i from $\overline{R}_{i-1}$ or add interesting ones from $\overline{Lr}_{i-1}$ (Sections 3.3 and 3.4) */
  $\overline{R}_i, \overline{Lr}_i \leftarrow column\_deletion(\gamma_i, \Lambda^{\epsilon_i}, \overline{R}_{i-1}, \overline{Lr}_{i-1})$;
  /* Check if possible not to do the step 3 (Section 3.3) */
  if $\gamma_i > \gamma_{i-1}$ **then**
    /* Step 3 (Section 2.5.3) starting the labelling algorithm with labels in $\overline{Lp}_{i-1}$) (T_{save} in Section 3.4) and limiting the size of $\overline{R}_i$ to MAX_COL (Section 3.5) */
    $\overline{R}_i^1, \overline{Lr}_i^1, \overline{Lp}_i \leftarrow route\_enumeration(\gamma_i, \Lambda^{\epsilon_i}, MAX\_COL, \overline{Lp}_{i-1})$;
    $\overline{R}_i \leftarrow \overline{R}_i^1 \cup \overline{R}_i$;
    $\overline{Lr}_i \leftarrow \overline{Lr}_i^1 \cup \overline{Lr}_i$;
  end
end
Return $\overline{R}_i$, $\overline{Lr}_i$, $\overline{Lp}_i$;

Algorithm 3: generate_route$(\gamma_i, \gamma_{i-1}, \Lambda^{\epsilon_{i-1}}, \Lambda^{\epsilon_i}, \overline{R_{i-1}}, \overline{Lr_{i-1}}, \overline{Lp_{i-1}})$.

used to evaluate the improvements induced by the techniques from Section 3. The second problem, the Bi-Objective Team Orienteering Problem with Time Windows (BOTOPTW), is a problem introduced in Parragh and Tricoire (2019). It allows the comparison of eCGEA with another algorithm from the literature.

The time limit is 4 hours for the BOVRPTW and 2 hours for the BOTOPTW. For each problem, we provide the computational times of the algorithms as well as their *performance profiles* Dolan and Moré (2002). The performance of an algorithm on an instance is defined as the ratio of its CPU time over the best CPU time among the compared algorithms. A *performance profile* represents the percentage of instances solved (y-axis) at a given performance (x-axis). For instance, a curve which passes through the point $(2, 0.8)$ indicates that 80% of the instances have been solved by the associated algorithm in less than twice the time of the fastest algorithm. Thus, a curve situated above another curve represents an algorithm with a better overall performance. If an instance is not solved by an algorithm, the CPU time for this instance and this algorithm is considered infinite. If an instance is not solved by any algorithm, then it is not added in the performance profile. The impact of each technique of Section 3 on SeM is evaluated in Appendix B for BOVRPTW and in Appendix D for BOTOPTW. In Appendix C, a compact formulation is embedded in the ϵ-constraint method to see the impact of eCGEA on BOVRPTW instances.

The remainder of the section is organized as follows. The two problems and the test instances are described in Section 4.1. The analysis on the BOVRPTW is reported in Section 4.2 and the one on the BOTOPTW in Section 4.3.

4.1. Test problems

Bi-Objective Vehicle Routing Problem With Time Windows. This problem is a straightforward extension of the Vehicle Routing Problem with Time Windows. Each customer in V' must be served a given quantity of goods by a vehicle defined by a capacity. Each edge is associated with two costs, and therefore a route has also two costs. Instances were created by combining two of Solomon's VRPTW instances[1] together. Let A and B be a couple of Solomon's instances with $|V| = 25$. We create an instance called $A - B$ as follows: the first location of clients, the time windows, the service time, the demand and the location of the depot are provided from file A, whereas the second location of clients are taken from file B. The cost matrix c^1 (respectively c^2) comes from the pairwise distance between the first (respectively second) locations of nodes. The distance between client a located at (x_a,y_a) and client b located at (x_b,y_b) is $\lfloor\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} + 1\rfloor$. Once all redundant resulting files are removed, it gives a total of 168 instances.

Bi-Objective Team Orienteering Problem with Time Windows. Parragh and Tricoire (2019) introduced and solved the Bi-Objective Team Orienteering Problem with Time Windows (BOTOPTW). It is a bi-objective problem based on the Team Orienteering problem (TOP) Gunawan et al. 2016. In the BOTOPTW, each $i \in V'$ is associated with profit p_i, a time window $[b_i; e_i]$ and a service time s_i. The profit is collected if i is visited during its time window. It is not necessary to visit all the nodes in V'. A travel time (or cost) c_{ij} is also associated with each edge. The first objective is to minimize the total travel time and the second objective is to maximize the collected profit. The BOTOPTW can be solved using eCGEA by setting $c^1_{ij} = c_{ij}$ and $c^2_{ij} = -p_i$. We use the 80 test instances used in Parragh and Tricoire 2019. These instances are derived from the Krolack instances Righini and Salani 2009 and are denoted $X - K - Z$ with X the Krolak instance it is based on, K the number of vehicles and Z the number of vertices.

4.2. Results on the BOVRPTW

The goal is to evaluate the contributions of the mechanisms from Section 3. The parameters used in the method are set as follows: the size of pre-defined ng-sets used for ng-route relaxation is $|N_i| = 8$, the parameters for the T_{dual} mechanisms are $W = \{0, 0.25, 0.5, 0.75, 1\}$ and $MAX\_COL = 30000$ (see Section 3.5). Upper bound sets have been pre-computed and given as an input to the algorithms such that the computational times are not impacted by the quality of the upper bound. An upper bound set is computed as follows. Model (3) is solved using aggregation weights $w_k \in \{0, 0.25, 0.5, 0.75, 1\}$ by means of a column generation algorithm. Then, a black-box MIP solver is embedded in an ϵ-constraint method to compute feasible solutions using the columns previously found.

Table 1 gives the characteristics of the instances. The instances are grouped into several types (*Type*). The type of an instance is defined by the types of the Solomon's instances used to build it. The table also reports the average number ($|ND|$) of non-dominated points and the average number ($|S|$) of supported points over the solved instances. The value *MGap* is the average gap $\gamma = 100 - \frac{c^1_{LB} * 100}{c^1_O}$ on the first objective between the lower bound LB and the optimal solution O for each ϵ computed by SeM. Finally, we report the total number of instances (*Number*), the number of instances solved by SEM and the number of instances solved by eCGEA. The results show that the

[1] Instances available on `https://www.sintef.no/nearp`

Table 1: Features of the instances of BOVRP.

Type	$\|ND\|$	$\|S\|$	MGap	Number	SeM	eCGEA
r1_c1	38	11	3.0	12	**12**	**12**
r1_c2	46	12	3.4	12	**11**	**11**
r1_rc	54	13	2.8	12	**12**	**12**
r2_c1	47	11	3.0	11	**1**	**1**
r2_c2	63	12	1.9	11	**1**	**1**
r2_rc	74	16	2.8	11	3	**6**
rc1_c1	18	8	3.9	8	**8**	**8**
rc1_c2	22	9	4.3	8	**8**	**8**
rc1_r	35	10	5.8	8	**8**	**8**
rc2_c1	44	8	3.4	8	1	**3**
rc2_c2	40	11	4.0	8	2	**4**
rc2_r	63	13	4.7	8	2	**3**
c1_c2	10	4	6.3	9	7	**8**
c1_rc	19	6	6.5	9	**8**	**8**
c1_r	37	10	6.7	9	8	**9**
c2_rc	38	14	2.5	8	1	**6**
c2_r	53	13	-	8	0	**2**
c2_c1	25	7	-	8	0	**7**
Total	18	12	3.79	168	93	**118**

mechanisms are able to improve the standard ϵ-constraint approach as an additional 25 instances are solved to optimality.

Now, we focus on the instances closed by at least one algorithm to evaluate more precisely the contribution of the techniques embedded in eCGEA. Tables 2 and 3 report the computation time in seconds (*Time*) for both methods as well as the computational time to generate the upper bound (*UB*). A dash (-) marks the fact that the algorithm was not able to converge. Except for T_{gap}, There are conditions to be checked before the application of a mechanism. We report the percentage of non-dominated points for which the conditions on each mechanisms are met ($X(\%)$ with X the mechanism name). For instance, if T_{valid} is equal to 17%, it means that the conditions not to use column generation were verified for 17% of the non-dominated set. The average on all these metrics on all instances solved by both methods are given in the line *Mean*. The table also indicates the number of instances solved optimally by each algorithm. In average, eCGEA is more than 50% faster than SeM. The speed up is less significant on the instances c1_c, c1_r and c1_rc. An explanation is that these instances have smaller non-dominated sets, and therefore SeM is iterated less often leaving less room for improvement for the proposed mechanisms.

All mechanisms are highly used during the algorithm: on average, T_{valid} is used between 27% and 42%, T_{save} between 19% and 32% and T_{dual} between 4% and 30%. The longer the time needed to solve an instance, the higher the use of a mechanism. This is especially true for T_{dual}. An explanation is that the instances that are the longest to solve are the ones where the procedure *route_enumeration* is also used the most.

Figure 3 represents the performance profiles of SeM and eCGEA. The profile of eCGEA is clearly better than the one of SeM. 20% of the instances are not closed by SeM, and eCGEA is

Table 2: Computational results of SeM and eCGEA on instances of type r_c, r_rc, rc_r and rc_c for $|V| = 25$.

	UB	SeM	eCGEA						UB	SeM	eCGEA				
Instance	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)	Instance	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)
r101_c1	1.8	4.9	**3.7**	57	17	43	0	rc101_c1	4.7	4.4	**4.1**	13	13	25	0
r101_c2	1.8	4.3	**3.3**	55	41	23	0	rc101_c2	6.1	4.8	**3.8**	50	43	7	0
r101_rc1	2.1	6.7	**4.7**	62	47	18	0	rc101_r1	10.0	12.2	**10.6**	46	27	31	0
r102_c1	25.0	102.2	**42.1**	74	16	63	0	rc102_c1	11.1	11.8	**9.3**	59	47	12	0
r102_c2	22.2	25.6	**19.5**	69	31	36	0	rc102_c2	15.3	18.0	**13.8**	33	21	21	0
r102_rc1	14.9	23.6	**16.7**	75	60	18	0	rc102_r1	30.8	30.1	**27.4**	69	37	33	0
r103_c1	21.8	24.9	**18.0**	64	61	11	0	rc103_c1	30.9	26.0	**20.8**	57	50	7	0
r103_c2	42.2	34.3	**27.9**	52	27	27	0	rc103_c2	59.5	71.8	**54.4**	46	26	26	0
r103_rc1	40.8	42.0	**30.7**	63	35	31	0	rc103_r1	26.9	29.4	**23.4**	57	43	16	0
r104_c1	60.8	42.7	**35.3**	43	23	26	0	rc104_c1	57.5	66.5	**54.6**	50	55	9	0
r104_rc1	178.8	1241.5	**264.7**	62	32	33	6	rc104_c2	25.9	20.0	**18.3**	43	21	36	0
r104_c2	65.0	75.4	**49.0**	57	30	27	0	rc104_r1	49.9	61.9	**51.2**	69	56	10	0
r105_c1	17.8	**16.9**	17.2	66	22	47	0	rc105_c1	14.7	13.9	**12.7**	71	33	29	0
r105_c2	15.6	14.7	**11.7**	58	23	39	0	rc105_c2	10.9	8.6	**7.3**	61	52	9	0
r105_rc1	8.5	13.0	**11.3**	60	29	34	0	rc105_r1	11.9	11.1	**9.1**	40	45	10	0
r106_c1	49.5	54.5	**40.4**	57	38	27	0	rc106_c1	14.7	8.3	**6.6**	40	33	13	0
r106_c2	88.8	61.7	**51.9**	54	30	29	0	rc106_c2	26.8	18.4	**15.9**	61	29	25	0
r106_rc1	95.0	189.2	**173.0**	58	33	28	0	rc106_r1	17.0	12.4	**11.0**	38	25	21	0
r107_c1	119.9	1322.7	**350.3**	66	56	24	10	rc107_c1	18.4	**12.1**	12.7	21	14	7	0
r107_c2	132.4	135.3	**103.4**	41	17	24	0	rc107_c2	37.0	20.9	**18.3**	52	36	20	0
r108_c1	52.6	52.1	**44.8**	44	18	29	0	rc107_r1	49.2	174.8	**115.3**	58	33	27	0
r108_c2	124.8	6960.1	**552.9**	48	35	27	17	rc108_c1	29.7	29.3	**26.3**	47	47	6	0
r109_c1	33.7	13.4	**11.5**	40	24	28	0	rc108_c2	31.2	31.7	**31.3**	45	30	20	0
r109_c2	22.0	24.7	**23.8**	54	36	21	0	rc108_r1	33.9	56.3	**39.2**	50	32	21	0
r109_rc1	20.1	26.6	22.9	60	33	31	0	rc201_c1	72.7	341.1	**215.9**	76	74	3	29
r110_c1	133.5	177.9	**159.7**	42	29	16	0	rc201_c2	61.1	70.6	**51.9**	56	52	11	0
r110_c2	82.9	69.5	**64.4**	36	19	22	0	rc201_r1	191.3	3740.5	**473.4**	62	52	27	27
r110_rc1	40.0	57.8	**44.6**	71	44	31	0	rc202_c2	292.5	-	**2674.0**	70	55	7	55
r111_c1	124.3	1152.1	**170.2**	61	17	43	4	rc205_c1	235.4	-	**3342.9**	74	63	14	63
r111_c2	143.7	199.3	**155.7**	41	27	20	0	rc205_c2	180.5	375.98	**188.1**	57	46	18	0
r111_rc1	86.8	135.1	**93.5**	62	39	29	0	rc205_r1	163.1	784.9	**272.3**	63	43	22	32
r112_c1	88.5	109.5	**83.6**	53	33	22	0	rc206_c1	341.7	-	**1450.9**	81	60	32	68
r112_rc1	64.1	242.6	**101.7**	38	22	30	3	rc206_c2	246.4	-	**3385.0**	54	59	11	32
r201_c1	385.4	1822.6	**1333.6**	65	63	20	59	rc206_r1	241.1	-	**1073.1**	66	45	28	47
r201_c2	246.5	1072.4	**730.9**	66	56	19	65								
r201_rc1	91.5	94.9	**81.6**	52	44	25	0								
r202_rc1	393.8	4989.1	**2044.5**	65	69	12	77								
r203_rc1	416.4	-	**4818.9**	86	31	3	30								
r205_rc1	176.9	1035.3	**649.8**	70	50	17	42								
r209_rc1	346.7	-	**11842.2**	62	72	11	85								
r210_rc1	379.5	-	**4303.5**	62	27	9	33								
Mean		570.3	**201.2**	57	35	28	7	Mean		209.2	**62.0**	51	38	18	3
Closed		38	**41**					Closed		29	**34**				

never 3 times longer than SeM.

4.3. Results on the BOTOPTW

The goal is to compare eCGEA with an algorithm from the literature proposed by Parragh and Tricoire (2019). The parameters used in the method are set as follows: the size of predefined ng-sets used for ng-route relaxation is $|N_i| = 8$, the parameters for the T_{dual} mechanisms

Table 3: Computational results of SeM and eCGEA on instances of type c_c, c_r and c_rc for $|V| = 25$.

	UB	SeM	eCGEA						UB	SeM	eCGEA				
Instance	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)	Instance	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)
c101_c2	25.5	**13.6**	19.2	40	20	40	0	c108_rc1	123.6	219.5	**170.4**	76	69	7	48
c101_rc1	14.8	13.5	**11.6**	50	50	17	0	c108_r1	553.4	-	**2961.9**	52	48	30	54
c101_r1	53.0	**35.0**	35.7	55	55	14	0	c109_c2	173.2	**270.3**	330.4	61	83	0	89
c102_c2	96.0	46.6	**39.2**	44	38	13	0	c109_rc1	159.9	**282.5**	352.7	74	66	11	26
c102_rc1	101.4	**112.0**	149.0	53	67	7	67	c109_r1	335.2	1642.0	**1171.1**	57	64	17	71
c102_r1	380.3	**872.3**	1049.5	51	53	17	40	c201_rc1	122.1	4349.7	**935.0**	29	43	21	71
c103_rc1	163.7	**184.3**	297.6	61	65	13	61	c201_r1	829.4	-	**2674.1**	65	76	13	91
c103_r1	354.3	4255.4	**796.5**	29	55	14	48	c201_c1	352.6	-	**419.7**	64	14	36	57
c104_c2	283.0	-	**5053.8**	43	39	22	39	c202_rc1	536.5	-	**1365.9**	50	61	6	61
c104_r1	641.7	2525.3	**1552.8**	64	48	23	26	c202_c1	748.9	-	**2539.4**	66	38	24	59
c105_c2	32.1	**47.2**	52.9	0	0	0	0	c203_c1	864.6	-	**10858.5**	50	67	13	79
c105_rc1	58.0	113.2	**59.5**	67	60	13	13	c204_rc1	1409.9	-	**10947.2**	65	68	10	75
c105_r1	139.9	**185.1**	500.8	44	72	12	72	c205_rc1	298.3	-	**571.2**	67	60	16	74
c106_c2	24.8	**26.1**	81.1	40	40	0	0	c205_r1	681.9	-	**5898.6**	59	70	25	80
c106_rc1	17.5	**11.0**	11.5	50	50	17	0	c205_c1	287.8	-	**768.9**	74	41	12	35
c106_r1	55.1	48.2	**45.8**	57	48	14	0	c206_rc1	269.2	-	**1003.0**	74	50	24	71
c107_c2	40.0	**48.1**	59.2	25	25	25	0	c206_c1	471.7	-	**7697.9**	68	50	18	59
c107_rc1	228.0	1782.6	**367.1**	73	73	13	80	c207_c1	635.8	-	**12257.1**	73	35	27	50
c107_r1	213.0	359.6	**291.4**	50	38	19	0	c208_rc1	421.2	-	**1591.6**	59	57	11	70
c108_c2	77.3	43.9	**41.4**	75	75	6	0	c208_c1	545.6	-	**878.0**	70	35	35	65
Mean		671.5	**315.9**	51	53	13	30								
Closed		24	**40**												

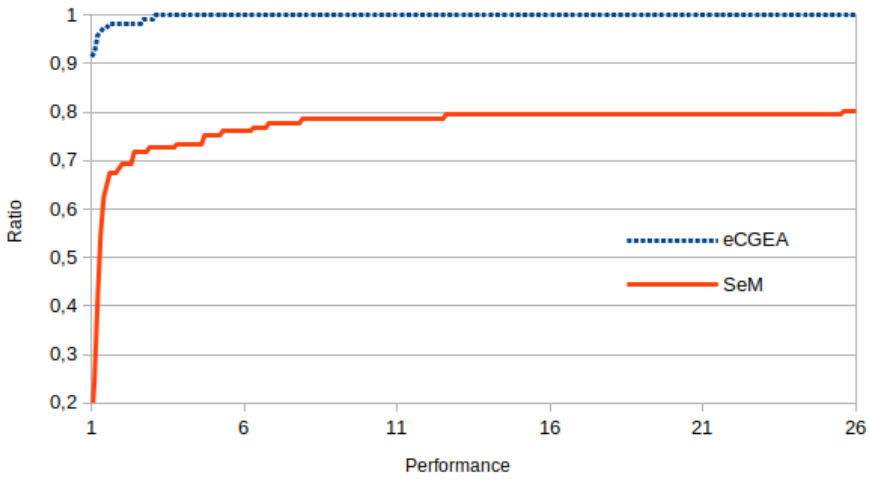


Figure 3: Performance profiles of SeM and eCGEA on BOVRPTW instances.

are $W = \{0, 1\}$ and $MAX\_COL = 30000$ (see Section 3.5). At each iteration i of eCGEA and SeM, an upper bound UB_i of $MP(\epsilon_i)$ (see Section 2.5.2) is computed as follows. The function *compute_lowerbound* returns the set of ng-routes $\widetilde{R}_i$ in the restricted linear master problem at the

end of the column generation (see Section 2.5.1). The non-elementary routes are removed from $\widetilde{R}_i$. The resulting integer program $RMP(\epsilon_i, \widetilde{R}_i)$ is then solved using a black box mixed integer linear programming (MILP) solver to obtain UB_i. The execution time of this heuristic is included in the execution time of eCGEA and SeM.

Tables 4 and 5 report the results on the 80 instances solved by Parragh and Tricoire (2019). The size of the non-dominated set ($|ND|$), the size of the supported set ($|S|$) and the average gap (MGap) between a non-dominated point and the closest lower bound for the first objective are reported. The computational times in seconds are given for Parragh and Tricoire's algorithm (B&B), SeM and eCGEA.The line *Mean* represents the average computation times for the instances closed by both methods. We also indicate the percentage of the non-dominated points found with each technique. As we do not use the same CPU than Parragh and Tricoire (2019) we checked the relative efficiency using the passmark CPU score[2]. Parragh and Tricoire (2019) use a 2.6-GHz Xeon E5-2650 v2 CPU with a CPU Single Thread Rating of 1691 and we use a 2.1-GHz Xeon E5-2695 v4 CPU with a CPU Single Thread Rating of 1628. As the two processors are close and the difference is in their favor, we did not modify their computational times by a factor of 1.04. They also implement the procedures in C++ and use Gurobi 6.5.0.

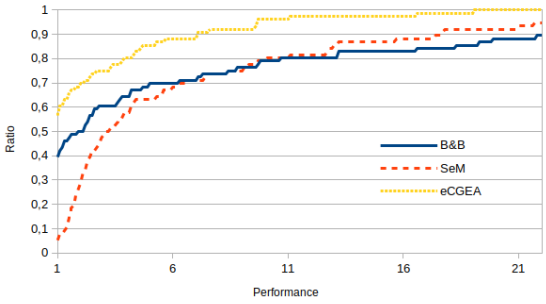
On the complete set of instances, eCGEA is almost 7 times faster than B&B, and it closes 6 additional instances. The techniques T_{lb} and T_{valid} are highly used for all instances. On the contrary, T_{dual} is less used. It means that the instances are well-adapted for column generation, and few columns are generated at each iteration when compared with BOVRPTW instances. Some $c101$ instances are solved faster by SeM than eCGEA. This is due to the fact that the mechanism T_{dual} is quite used on these instances, and it can slow down the solution if the number of columns to be found during the enumeration is just a little bit greater than the MAX_COL parameter. An additional remark, shown in Appendix D, is that the mechamism T_{valid} is crucial for the efficiency of eCGEA.

Figure 4 represents the performance profiles of B&B, SeM and eCGEA. First, the profiles show again that on the whole set of instances, eCGEA dominates the other methods. However, this observation should be mitigated. On instances such as those in classes $c101$, $rc101$ and $pr01$, eCGEA is much more effective as it is considerably faster, and it is able to find new non-dominated sets. On other instances such as those in the class $r101$, B&B is strictly more efficient than our algorithms. It should be noted that the worst computational time of eCGEA on this class of instances is only 70 seconds. Even without the speed-up mechanisms, SeM is competitive with B&B. This underlines the fact that the choice to base the method on a state-of-the-art single objective column generation algorithm is important. However, advanced column generation techniques can add unnecessary additional time required to solve some instances like the $r101$ instances.

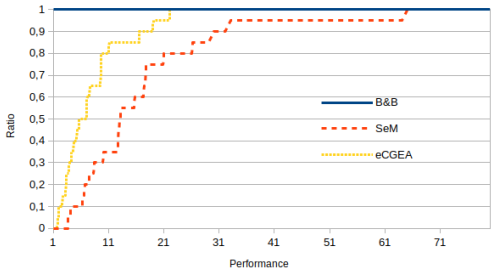
5. Conclusion

In this paper, we propose an exact algorithm to solve bi-objective vehicle routing problems. The idea is to use a state-of-the-art column generation-and-enumeration algorithm for single objective vehicle routing problems in an ϵ-constraint approach to obtain an easy to implement method. We provide different procedures and mechanisms to improve the efficiency of the algorithm. It should

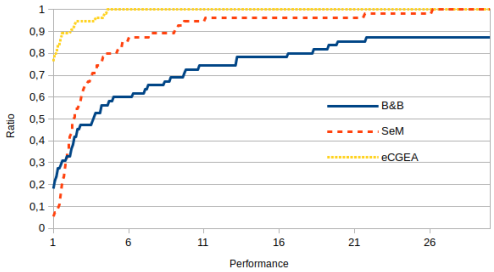
[2] https://www.cpubenchmark.net/cpu_list.php



(a) On all instances.



(b) On the $r101$ instances.



(c) On the $c101$, $rc101$, $pr01$ instances.

Figure 4: Performance profiles of B&B, SeM and eCGEA on BOVRPTW instances.

be noted that these procedures rely on the properties of the column generation-and-enumeration algorithm as well as observations on the objective space. They are not specific to vehicle routing problems and could be used for other problems. The efficiency of the improvement techniques have been tested on the Bi-Objective Vehicle Routing Problem allowing the algorithm to outperform a direct application of the ϵ-constraint method. The results can be used to benchmark other algorithms for this problem which is representative of bi-objective vehicle routing problems. We have also tested our algorithm on the Bi-Objective Team Orienteering Problem with Time Windows. The algorithm outperforms the exact algorithm proposed in the literature so far. This study also presents the limits of using the ϵ-constraint method for bi-objective vehicle routing problems as the gap increases with smaller ϵ values. A perspective could be to construct pertinent cuts to reduce this gap.

References

Y. P. Aneja and K. P. Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

R. Arab, S. Ghaderi, and R. Tavakkoli-Moghaddam. Solving a new multi-objective inventory-routing problem by a non-dominated sorting genetic algorithm. *International Journal of Engineering-Transactions A: Basics*, 31(4): 588–596, 2018.

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.

N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective integer programming: The balanced box method. *INFORMS Journal on Computing*, 27(4):735–754, 2015a.

N. Boland, H. Charkhgard, and M. Savelsbergh. A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method. *INFORMS Journal on Computing*, 27(4):597–618, 2015b.

G. A. Bula, H. M. Afsar, F. A. González, C. Prodhon, and N. Velasco. Bi-objective vehicle routing problem for hazardous materials transportation. *Journal of Cleaner Production*, 206:976–986, 2019.

V. Chankong and Y. Haimes. Multiobjective decision making: theory and methodology, 1983.

J.-F. Cordeau. *The VRP with time windows*. Groupe d'études et de recherche en analyse des décisions Montréal, 2000.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

E. Demir, T. Bektaş, and G. Laporte. The bi-objective pollution-routing problem. *European Journal of Operational Research*, 232(3):464–478, 2014.

E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

E. Glize, R. Roberti, N. Jozefowiez, and S. U. Ngueveu. Exact methods for mono-objective and bi-objective multi-vehicle covering tour problems. *European Journal of Operational Research*, 283(3):812–824, 2020.

A. Gunawan, H. C. Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

Y. Haimes, L. S. Lasdon, and D. Da Wismer. 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.

E. E. Halvorsen-Weare and M. W. Savelsbergh. The bi-objective mixed capacitated general routing problem with different route balance criteria. *European Journal of Operational Research*, 251(2):451–465, 2016.

P. Hansen. Bicriterion path problems. In *Multiple criteria decision making theory and application*, pages 109–127. Springer, 1980.

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

N. Jozefowiez, F. Semet, and E.-G. Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2):293–309, 2008.

A. A. Kovacs, S. N. Parragh, and R. F. Hartl. The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research*, 247(2):441–458, 2015.

N. Labadie and C. Prodhon. A survey on multi-criteria analysis in logistics: Focus on vehicle routing problems. In *Applications of Multi-Criteria and Game Theory Approaches*, pages 3–29. Springer, 2014.

P. Matl, R. F. Hartl, and T. Vidal. Workload equity in vehicle routing: The impact of alternative workload resources. *Computers & Operations Research*, 110:116–129, 2019.

J. C. Molina, I. Eguia, J. Racero, and F. Guerrero. Multi-objective vehicle routing problem with cost and emission functions. *Procedia-Social and Behavioral Sciences*, 160:254–263, 2014.

S. N. Parragh and F. Tricoire. Branch-and-bound for bi-objective integer programming. *INFORMS Journal on Computing*, 31(4):805–822, 2019.

A. Przybylski and X. Gandibleux. Multi-objective branch and bound. *European Journal of Operational Research*, 260(3):856–872, 2017.

P. Reiter and W. J. Gutjahr. Exact hybrid algorithms for solving a bi-objective vehicle routing problem. *Central European Journal of Operations Research*, 20(1):19–43, 2012.

G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4): 1191–1203, 2009.

J. Sáez-Aguado and P. C. Trandafir. Variants of the ϵ-constraint method for biobjective integer programming problems: application to p-median-cover problems. *Mathematical Methods of Operations Research*, 87(2):251–283, 2018.

E. M. Toro, J. F. Franco, M. G. Echeverri, and F. G. Guimarães. A multi-objective model for the green capacitated location-routing problem considering environmental impact. *Computers & Industrial Engineering*, 110:114–125, 2017.

P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

F. Tricoire. Multi-directional local search. *Computers & Operations Research*, 39(12):3089–3101, 2012.

E. L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1995.

C. A. Vega-Mejía, J. R. Montoya-Torres, and S. M. Islam. Consideration of triple bottom line objectives for sustainability in the optimization of vehicle routing and loading operations: a systematic literature review. *Annals of Operations Research*, pages 1–65, 2017.

H. Yu and W. Solvang. An improved multi-objective programming with augmented ε-constraint method for hazardous waste location-routing problems. *International Journal of Environmental Research and Public Health*, 13(6):548, 2016.

Table 4: Computational experiments of the BOTOPTW on the instances $c101$ and $pr01$.

Instance characteristics				B&B	SeM	eCGEA				
Instance	$\|ND\|$ (-)	$\|NS\|$ (-)	MGap (%)	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)
c101_1_15	18	10	24.9	9.8	11.9	**7.8**	61	61	6	0
c101_1_20	20	10	24.5	44.9	23.3	**9.7**	60	70	0	0
c101_1_25	21	9	21.7	94.9	30.8	**10.8**	67	67	5	0
c101_1_30	21	9	21.7	197.2	37.4	**15.1**	67	67	5	0
c101_1_35	23	9	20.5	263.9	47.7	**16.0**	70	70	4	0
c101_2_15	24	13	20.6	44.9	28.9	**16.6**	58	58	4	0
c101_2_20	32	16	17.8	755.3	56.8	**38.0**	56	62	0	0
c101_2_25	37	20	14.9	5931.0	154.7	**96.3**	51	54	3	3
c101_2_30	37	20	14.9	-	135.2	**95.0**	54	57	3	3
c101_2_35	41	18	13.4	-	78.2	**49.9**	58	61	7	0
c101_3_15	24	13	20.6	40.7	25.7	**18.7**	58	58	4	0
c101_3_20	33	16	17.4	844.4	**356.1**	442.1	58	54	0	27
c101_3_25	42	21	13.7	-	2712.7	**2193.3**	57	52	2	26
c101_3_30	-	-	-	-	-	-	-	-	-	-
c101_3_35	-	-	-	-	-	-	-	-	-	-
c101_4_15	24	13	20.6	44.3	27.7	**18.5**	58	58	4	0
c101_4_20	33	16	17.4	838.7	**372.8**	477.6	58	51	3	27
c101_4_25	42	21	13.7	-	**4110.8**	4331.5	57	52	2	26
c101_4_30	-	-	-	-	-	-	-	-	-	-
c101_4_35	-	-	-	-	-	-	-	-	-	-
Mean	26	13	20.2	759.2	97.8	**97.2**	60	61	3	5
Closed				12	**16**	**16**				
pr01_1_15	26	7	20.3	**4.0**	22.3	9.4	73	61	11	0
pr01_1_20	39	5	15.5	19.6	68.8	**12.4**	85	85	5	0
pr01_1_25	72	7	9.6	135.6	128.2	**21.5**	90	85	6	0
pr01_1_30	93	7	8.2	1222.6	1468.8	**56.2**	91	85	7	0
pr01_1_35	85	9	0.0	3345.3	1678.2	**173.3**	88	82	11	0
pr01_2_15	36	8	15.5	**3.4**	25.0	12.6	78	69	11	0
pr01_2_20	72	13	11.8	**36.9**	126.9	78.5	79	69	10	1
pr01_2_25	121	13	7.1	341.4	2905.7	**134.1**	84	81	5	0
pr01_2_30	153	10	6.5	1800.9	757.8	**476.7**	89	84	8	2
pr01_2_35	-	-	-	-	-	-	-	-	-	-
pr01_3_15	37	8	15.2	**2.7**	29.8	12.2	76	70	11	0
pr01_3_20	67	10	12.6	46.6	68.0	**32.7**	79	69	10	0
pr01_3_25	134	11	6.5	201.6	202.5	**56.8**	92	87	4	0
pr01_3_30	169	10	5.9	1297.0	551.6	**178.5**	93	89	5	0
pr01_3_35	195	15	4.3	-	2865.5	**1657.0**	89	81	9	1
pr01_4_15	37	8	15.2	**2.7**	25.1	11.8	76	70	11	0
pr01_4_20	67	11	12.6	40.2	67.2	**32.3**	79	69	12	0
pr01_4_25	135	11	6.4	166.6	236.6	**45.5**	92	89	3	0
pr01_4_30	169	10	5.9	1175.0	472.8	**121.3**	93	90	5	0
pr01_4_35	195	15	4.3	-	2530.0	**2341.5**	88	82	10	1
Mean	89	9	10	578.9	519.7	**86.2**	85	78	8	0
Closed				17	**19**	**19**				

Table 5: Computational experiments of the BOTOPTW on the instances $r101$ and $rc101$.

Instance characteristics				B&B	SeM	eCGEA				
Instance	$\|ND\|$ (-)	$\|NS\|$ (-)	MGap (%)	Time (s)	Time (s)	Time (s)	T_{lb} (%)	T_{valid} (%)	T_{save} (%)	T_{dual} (%)
r101_1_15	9	3	29.3	**0.3**	9.0	5.0	56	56	22	0
r101_1_20	10	5	28.2	**0.2**	12.9	4.4	70	80	10	0
r101_1_25	10	5	28.2	**0.5**	16.3	5.5	70	80	10	0
r101_1_30	16	3	24.2	**1.6**	20.3	6.9	81	81	6	0
r101_1_35	23	4	17.7	**3.0**	38.4	15.8	83	74	9	4
r101_2_15	25	6	14.8	**0.6**	15.7	11.4	76	44	32	0
r101_2_20	27	8	15.1	**1.3**	23.1	12.5	78	68	18	0
r101_2_25	32	6	14.8	**2.0**	42.0	14.1	84	72	16	0
r101_2_30	40	7	12.5	**6.3**	52.9	21.1	80	75	7	0
r101_2_35	47	7	10.8	**10.1**	74.8	38.8	85	62	23	0
r101_3_15	45	8	9.9	**1.6**	28.3	15.3	82	60	22	0
r101_3_20	44	9	10.7	**2.5**	39.1	18.9	79	64	18	0
r101_3_25	54	10	10.2	**4.5**	58.5	25.3	83	70	15	0
r101_3_30	62	10	9.1	**12.5**	78.4	40.1	84	63	22	0
r101_3_35	73	11	8.0	**30.3**	111.8	59.3	84	63	22	0
r101_4_15	51	9	9.1	**1.9**	32.9	18.4	82	61	22	0
r101_4_20	56	10	9.0	**3.6**	47.3	25.2	84	64	21	0
r101_4_25	65	11	9.2	**7.0**	70.3	32.7	83	72	14	0
r101_4_30	77	12	8.2	**18.2**	120.5	48.0	82	74	12	0
r101_4_35	102	11	6.7	**37.9**	157.6	68.2	87	68	19	0
Mean	43	8	14.3	**7.3**	52.5	24.3	80	67	17	0
rc101_1_15	11	6	25.3	**1.1**	10.0	2.8	73	82	0	0
rc101_1_20	11	6	25.5	**2.4**	10.5	3.7	73	82	0	0
rc101_1_25	12	6	24.4	4.8	18.1	**4.2**	75	83	0	0
rc101_1_30	12	6	24.4	5.4	21.5	**5.2**	75	83	0	0
rc101_1_35	12	6	24.4	7.0	24.9	**6.5**	75	83	0	0
rc101_2_15	25	10	15.0	**6.2**	19.9	9.1	72	64	12	0
rc101_2_20	27	10	14.5	22.4	29.2	**12.3**	74	67	11	0
rc101_2_25	29	9	13.9	48.2	31.7	**11.7**	79	72	10	0
rc101_2_30	29	9	13.9	57.6	40.1	**13.9**	79	72	10	0
rc101_2_35	29	9	13.9	81.6	44.1	**16.4**	79	72	10	0
rc101_3_15	25	10	15.1	**8.5**	18.5	9.0	72	64	12	0
rc101_3_20	34	11	12.5	35.1	30.2	**16.1**	74	65	15	0
rc101_3_25	46	14	10.0	180.2	50.8	**25.7**	76	65	13	0
rc101_3_30	46	14	10.0	220.4	59.8	**26.3**	76	65	13	0
rc101_3_35	46	14	10.0	332.1	70.0	**31.3**	76	65	13	0
rc101_4_15	25	10	15.1	**6.9**	15.6	9.9	72	64	12	0
rc101_4_20	36	13	12.2	46.1	33.0	**18.0**	75	64	17	0
rc101_4_25	49	16	9.9	262.9	54.2	**27.0**	75	61	18	0
rc101_4_30	53	16	9.6	478.6	71.0	**36.5**	74	64	15	0
rc101_4_35	59	17	8.4	946.2	93.3	**52.0**	76	63	15	0
Mean	31	11	15.4	137.7	37.3	**16.9**	75	70	10	0

Appendix A. Optimal solution of MP(ϵ)

This appendix proves that it is valid to use `GENROUTE` to find an optimal solution of MP(ϵ).

Theorem 2. *The notations have been introduced in Section 2.2. The linear relaxation of MP(ϵ) is as follows:*

$$\begin{cases} \text{minimize} \sum_{r\in R} c_r^1 x_r & & (A.1.1) & \\ \text{s.t.} \sum_{r\in R} a_{ir} x_r \geq d_i & i \in V' & (A.1.2) & \longleftrightarrow \lambda_i^\epsilon \\ \sum_{r\in R} x_r \leq |K| & & (A.1.3) & \longleftrightarrow \lambda_0^\epsilon \\ \sum_{r\in R} c_r^2 x_r \leq \epsilon & & (A.1.4) & \longleftrightarrow \alpha^\epsilon \\ x_r \geq 0 & r \in R & (A.1.5) & \end{cases} \tag{A.1}$$

Let $\lambda_i^\epsilon \geq 0$ ($i \in V'$), $\lambda_0^\epsilon \leq 0$ and $\alpha^\epsilon \leq 0$ be the duals associated with constraints (A.1.2), (A.1.3) and (A.1.4). The dual of the formulation (A.1) is as follows:

$$\begin{cases} \text{maximize} \sum_{i\in V'} d_i \lambda_i^\epsilon + |K|\lambda_0^\epsilon + \epsilon\alpha^\epsilon & & (A.2.1) \\ \text{s.t.} \sum_{i\in V'} a_{ir}\lambda_i^\epsilon + \lambda_0^\epsilon + c_r^2\alpha^\epsilon \leq c_r^1 & r \in R & (A.2.2) \\ \lambda_i^\epsilon \geq 0 & i \in V' & (A.2.3) \\ \lambda_0^\epsilon \leq 0 & & (A.2.4) \\ \alpha^\epsilon & & (A.2.5) \end{cases} \tag{A.2}$$

Let x^{LB} be the optimal solution of the formulation (A.1) and x^{UB} be a feasible solution of MP(ϵ). Let $\Lambda^{\epsilon\ LB} = (\lambda^{\epsilon\ LB}, \alpha^{\epsilon\ LB})$ be the optimal solution of the formulation (A.2).

The sum of the reduced cost of routes $r \in R$ such that $x_r^{UB} > 0$ is less than or equal to the difference of cost c^1 between x^{UB} and x^{LB}. This difference of cost is called the gap γ.

Proof. The gap is described in Equation (A.3) and the reduced cost of a route r in Equation (A.4).

$$\gamma = \sum_{r\in R} c_r^1 x_r^{UB} - \sum_{r\in R} c_r^1 x_r^{LB} \tag{A.3}$$

$$\overline{c_r} = c_r^1 - \sum_{i\in V'} a_{ir}\lambda_i^{\epsilon\ LB} - \lambda_0^{\epsilon\ LB} - c_r^2\alpha^{\epsilon\ LB} \qquad \forall r \in R \tag{A.4}$$

By (A.1.2) and (A.2.3), $\forall i \in V'$, $\sum_{r\in R} a_{ir} x_r^{UB} \lambda_i^{\epsilon\ LB} \geq d_i \lambda_i^{\epsilon\ LB}$.

$$\text{Thus,} \sum_{i\in V'}\sum_{r\in R} a_{ir} x_r^{UB} \lambda_i^{\epsilon\ LB} \geq \sum_{i\in V'} d_i \lambda_i^{\epsilon\ LB} \tag{A.5}$$

$$\text{By (A.1.3) and (A.2.4),} \sum_{r\in R} x_r^{UB} \lambda_0^{\epsilon\ LB} \geq |K|\lambda_0^{\epsilon\ LB} \tag{A.6}$$

$$\text{Finally, by (A.1.4) and (A.2.5),} \sum_{r\in R} c_r^2 x_r^{UB} \alpha^{\epsilon\ LB} \geq \epsilon\alpha^{\epsilon\ LB} \tag{A.7}$$

By the strong duality theorem:

$$\sum_{r\in R} c_r^1 x_r^{LB} = \sum_{i\in V'} d_i \lambda_i^{\epsilon\ LB} + |K|\lambda_0^{\epsilon\ LB} + \epsilon\alpha^{\epsilon\ LB}$$

$$\Rightarrow \sum_{r\in R} c_r^1 x_r^{LB} \leq \sum_{i\in V'}\sum_{r\in R} a_{ir} x_r^{UB} \lambda_i^{\epsilon\ LB} + \sum_{r\in R} x_r^{UB}\lambda_0^{\epsilon\ LB} + \sum_{r\in R} c_r^2 x_r^{UB}\alpha^{\epsilon\ LB} \text{ by (A.5), (A.6) et (A.7)}$$

$$\Rightarrow \sum_{r\in R} c_r^1 x_r^{UB} - \sum_{r\in R} c_r^1 x_r^{LB} \geq \sum_{r\in R} c_r^1 x_r^{UB} - \sum_{i\in V'}\sum_{r\in R} a_{ir} x_r^{UB}\lambda_i^{\epsilon\ LB} - \sum_{r\in R} x_r^{UB}\lambda_0^{\epsilon\ LB} - \sum_{r\in R} c_r^2 x_r^{UB}\alpha^{\epsilon\ LB}$$

$$\Rightarrow \gamma \geq \sum_{r\in R}(c_r^1 - \sum_{i\in V'} a_{ir}\lambda_i^{\epsilon\ LB} - \lambda_0^{\epsilon\ LB} - c_r^2\alpha^{\epsilon\ LB}) x_r^{UB} \text{ by (A.3)}$$

$$\Rightarrow \gamma \geq \sum_{r\in R} \overline{c_r} x_r^{UB} \text{ by (A.4)}$$

Therefore, the sum of the reduced cost of the routes such that $x_r^{UB} > 0$ is less than or equal to the gap. □

Let LB and UB be a lower bound and an upper bound of MP(ϵ) and $\Lambda^{\epsilon\ LB}$ be the dual solution associated with LB. In the following paragraph, we consider that the reduced costs are computed with respect to $\Lambda^{\epsilon\ LB}$. The step 3 of `GENROUTE` (see Section 2.5.3) generates all routes with reduced cost less than or equal to the gap $\gamma = c^1(UB) - c^1(LB)$. Let x_P be a feasible solution associated with a non-dominated point P in the objective space such that $c^1(LB) \leq c^1(P) \leq c^1(UB)$ and $c^2(P) \leq \epsilon$. By Theroem 2, x_P is composed by routes with reduced cost less than or equal to $c^1(P) - c^1(LB) \leq \gamma$. These routes are generated in a set of routes $\overline{R}$ by the route enumeration of `GENROUTE` between LB and UB. Thus, any solution x_P can be found in $\overline{R}$ by modifying the ϵ value of RMP(ϵ,$\overline{R}$).

Table B.6: Computational results of SeM and SeM with each technique on instances of type r_c, r_rc, rc_r and rc_c for $|V| = 25$.

Instance	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}		SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}
r101_c1	4.9	4.8	4.8	4.2	4.7	**3.6**	rc101_c1	4.4	4.1	4.4	4.2	**3.5**	3.8
r101_c2	4.3	4.4	4.4	**3.6**	4.1	**3.6**	rc101_c2	4.8	5.3	5.3	4.3	4.6	**4.1**
r101_rc1	6.7	6.3	6.6	5.2	5.4	**4.8**	rc101_r1	12.2	12.5	12.9	**11.2**	11.7	**11.2**
r102_c1	102.2	90.2	88.8	96.9	69.9	**42.7**	rc102_c1	11.8	11.3	10.7	10.4	**9.8**	10.5
r102_c2	25.6	24.6	23.0	22.7	24.3	**21.6**	rc102_c2	18.0	19.1	**16.7**	**16.7**	16.8	17.5
r102_rc1	23.6	23.7	22.7	20.6	20.5	**19.2**	rc102_r1	30.1	30.4	30.4	28.0	28.9	**26.8**
r103_c1	24.9	26.3	21.4	22.7	24.4	**21.5**	rc103_c1	26.0	25.9	25.3	22.7	**21.8**	22.3
r103_c2	34.3	34.4	32.7	33.8	**29.4**	30.1	rc103_c2	71.8	74.4	82.6	67.4	72.4	**53.3**
r103_rc1	42.0	41.2	38.4	39.2	**33.1**	37.9	rc103_r1	29.4	29.7	26.9	26.2	**22.1**	26.4
r104_c1	42.7	42.7	**39.7**	42.6	41.8	40.1	rc104_c1	66.5	67.4	77.2	**60.3**	66.9	76.1
r104_rc1	1241.5	1269.1	1440.0	1268.3	**139.1**	288.5	rc104_c2	20.0	20.9	21.3	21.1	**16.3**	19.4
r104_c2	75.4	76.1	64.8	71.8	71.7	**62.5**	rc104_r1	61.9	59.0	57.4	**53.5**	61.0	55.3
r105_c1	16.9	16.5	17.7	16.2	16.0	**15.4**	rc105_c1	13.9	13.2	13.2	**12.2**	13.6	12.4
r105_c2	14.7	14.0	13.3	12.6	14.3	**11.9**	rc105_c2	8.6	8.7	9.2	7.3	8.2	**6.9**
r105_rc1	13.0	13.2	13.1	11.8	12.3	**10.6**	rc105_r1	11.1	10.7	9.6	**9.3**	10.2	9.8
r106_c1	54.5	50.9	46.7	50.1	**42.1**	46.0	rc106_c1	8.3	7.9	8.0	7.4	7.9	**7.0**
r106_c2	61.7	63.3	60.5	60.1	58.3	**55.4**	rc106_c2	18.4	17.8	17.5	**16.5**	17.1	16.7
r106_rc1	189.2	184.9	175.5	184.9	**149.4**	181.8	rc106_r1	12.4	13.3	12.7	12.1	**11.8**	11.9
r107_c1	1322.7	1330.4	1194.5	1107.2	**196.6**	1201.1	rc107_c1	12.1	13.1	12.6	12.1	**11.9**	12.8
r107_c2	135.3	136.7	128.5	127.1	125.1	**122.1**	rc107_c2	20.9	20.1	20.1	18.8	18.6	**18.5**
r108_c1	52.1	50.4	**47.2**	51.1	51.1	48.3	rc107_r1	174.8	155.5	138.0	144.1	170.8	**118.0**
r108_c2	6960.1	6356.7	8150.9	6993.9	**261.7**	1242.9	rc108_c1	29.3	28.9	27.4	**26.9**	27.6	27.3
r109_c1	13.4	13.3	13.4	12.7	13.9	**12.0**	rc108_c2	31.7	33.6	32.2	32.3	30.7	**29.5**
r109_c2	24.7	23.6	26.3	22.7	22.6	**19.9**	rc108_r1	56.3	49.5	61.3	47.1	49.7	**39.2**
r109_rc1	26.6	32.5	26.4	25.3	26.9	**25.2**	rc201_c1	341.1	326.4	364.2	224.5	**155.6**	201.4
r110_c1	177.9	177.0	195.6	**165.7**	177.4	172.4	rc201_c2	70.6	65.3	80.0	52.9	54.8	**50.6**
r110_c2	69.5	66.3	68.1	71.6	65.1	**63.0**	rc201_r1	3740.5	3377.1	3430.4	2489.2	**338.6**	1229.0
r110_rc1	57.8	61.6	52.7	54.4	56.3	**47.3**	rc202_c2	-	-	-	-	**3527.9**	5996.7
r111_c1	1152.1	1065.2	1110.4	1105.0	**171.9**	1106.6	rc205_c1	-	-	-	-	13129.3	**12349.6**
r111_c2	199.3	177.5	182.0	190.4	184.4	**153.0**	rc205_c2	376.0	324.6	351.3	258.2	272.8	**212.3**
r111_rc1	135.1	126.9	140.0	110.9	**106.2**	111.9	rc205_r1	784.9	753.4	641.3	555.6	**279.8**	313.0
r112_c1	109.5	107.2	105.6	96.1	101.8	**93.6**	rc206_c1	-	-	-	-	**1507.8**	-
r112_rc1	242.6	220.9	208.9	198.1	142.6	**107.7**	rc206_c2	-	-	-	-	**298.7**	-
r201_c1	1822.6	1592.8	2121.0	1547.4	**671.1**	1172.5	rc206_r1	-	-	-	-	**1238.9**	-
r201_c2	1072.4	954.1	937.8	626.4	**503.7**	514.3							
r201_rc1	94.9	86.8	90.1	89.1	92.5	**86.0**							
r202_rc1	4989.1	3840.9	3784.2	4573.4	2866.6	**1440.8**							
r205_rc1	1035.3	858.6	1327.7	735.8	**497.9**	588.0							
r210_rc1	-	-	-	-	**7138.7**	-							
Mean	570.3	507.0	579.6	522.9	**186.7**	242.8	Mean	209.2	192.4	193.1	146.6	62.6	91.1
Closed	38	38	38	38	**39**	38	Closed	24	24	24	24	**29**	26

Appendix B. Results on the BOVRPTW

This appendix underlines the impact of each technique of Section 3 embedded SeM on BOVRPTW instances. Tables B.6 and B.7 provides the computation times in seconds (*Time*) for SeM and SeM improved with each technique. The column called T_{gap} (respectively T_{lb}, T_{valid}, T_{save} and T_{dual}) represents the algorithm SeM improved with the technique T_{gap} (respectively T_{lb}, T_{valid}, T_{save} and T_{dual}). The line *Mean* represents the average computation times for the instances closed by all methods, and the line *Closed* represents the number of closed instances. Furthermore, Figure B.5 represents the performance profiles of SeM and SeM improved with each technique.

Thanks to the mean CPU time and the performance profiles, we can see that all techniques except T_{lb} provide improvements to SeM. The technique T_{save} is clearly the most impacting improvement, and it helps to close 17 instances when compared to SeM. The technique T_{lb} can improve SeM (like on $rc201\_r1$ and $c104\_rc1$), but it can

Table B.7: Computational results of SeM and eCGEA on instances of type c_c, c_r and c_rc for $|V| = 25$.

Instance	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}		SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}
c101_c2	13.6	**12.2**	72.9	13.5	13.2	13.2	c108_c2	43.9	38.1	42.5	**34.9**	43.3	35.5
c101_rc1	13.5	13.0	12.9	13.1	**12.8**	12.9	c108_rc1	219.5	209.6	176.1	**141.6**	187.4	159.4
c101_r1	35.0	33.3	42.9	30.8	33.0	**30.7**	c108_r1	-	11746.7	-	11379.5	**1569.8**	11436.9
c102_c2	46.6	**42.1**	43.3	42.8	42.3	45.4	c109_c2	270.3	214.1	261.1	**199.4**	311.9	245.2
c102_rc1	112.0	103.5	105.9	**79.8**	117.2	179.9	c109_rc1	282.5	262.2	280.8	**257.0**	285.3	281.6
c102_r1	872.3	812.7	861.5	804.8	**721.4**	942.0	c109_c2	270.3	214.1	261.1	**199.4**	311.9	245.2
c103_rc1	184.3	185.4	144.0	**135.8**	181.5	193.7	c201_rc1	4349.7	2498.5	3558.6	1814.8	**1006.5**	1583.8
c103_r1	4255.4	3533.2	4691.6	3788.5	**755.5**	4067.5	c201_r1	-	-	-	-	**2287.1**	-
c104_c2	-	-	-	-	**2934.8**	-	c201_c1	-	-	-	-	**411.6**	11589.8
c104_rc1	-	13175.9	12583.2	**12341.7**	13719.9	-	c202_c1	-	-	-	-	**2050.3**	-
c104_r1	2525.3	2711.5	2221.5	2448.1	**1671.6**	2319.4	c203_c1	-	-	-	-	**8439.1**	-
c105_c2	47.2	**38.0**	50.4	44.2	48.9	39.8	c205_rc1	-	12394.0	5202.8	7838.8	**383.0**	6714.9
c105_rc1	113.2	201.5	115.1	**73.2**	106.3	78.7	c205_r1	-	-	-	-	**12841.8**	-
c105_r1	185.1	220.5	224.7	180.6	**148.9**	181.2	c205_c1	-	-	-	-	**796.7**	-
c106_c2	26.1	**23.1**	71.4	24.6	25.9	24.2	c206_rc1	-	-	-	-	**834.2**	-
c106_rc1	11.0	**10.2**	11.4	10.5	10.5	10.8	c206_c1	-	-	-	-	**7082.7**	-
c106_r1	48.2	43.4	50.3	47.3	45.2	**42.3**	c207_c1	-	-	-	-	**9578.0**	-
c107_c2	48.1	47.6	51.3	**47.1**	48.6	54.0	c208_rc1	-	-	-	-	**8982.4**	-
c107_rc1	1782.6	1855.0	2625.2	1649.8	**681.8**	2304.2	c208_c1	-	-	-	-	**605.9**	-
c107_r1	359.6	329.4	340.7	337.2	358.1	**329.0**							
Mean	671.5	568.8	679.9	517.5	**298.7**	559.2							
Closed	24	27	26	27	**39**	27							

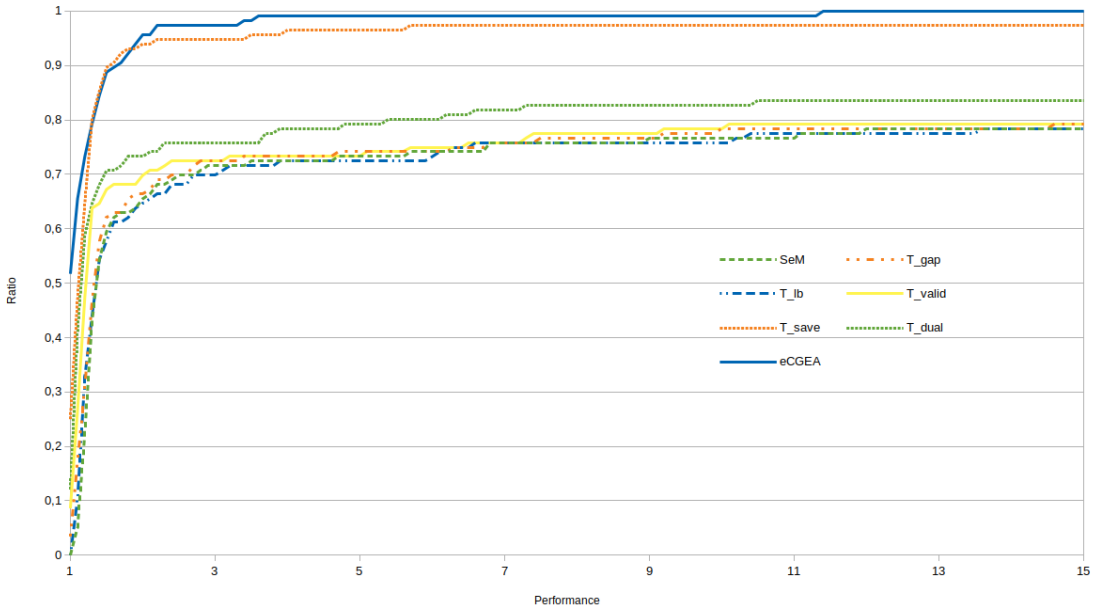


Figure B.5: Performance profiles of SeM and SeM improved with each technique on BOVRPTW instances.

also slow down it (like on $r104\_rc1$ and $r108\_c2$). Indeed, the column generation algorithm is too slow for some

instances, and T_{lb} is conceived to remove this step. Moreover, the set covering problems, master problems of column generation for VRP, are highly degenerated and present multiple optimal dual solutions.

Appendix C. Results on the BOVRPTW

This appendix provides the results of solving a multi-commodity network flow model with time window and capacity constraints Cordeau 2000 embedded in an ϵ-constraint technique (*compact_formulation*) on the BOVRPTW instances. Table C.8 provides the computation times in seconds (*Time*) for eCGEA and *compact_formulation*. The line *Mean* represents the average computation times for the instances closed by all methods, and the line *Closed* represents the number of closed instances.

Table C.8: Computational results of eCGEA and *compact_formulation* on instances of type *c_c*, *c_r* and *c_rc* for $|V| = 25$.

	eCGEA	*compact_formulation*		eCGEA	*compact_formulation*
Instance	Time (s)	Time (s)	Instance	Time (s)	Time (s)
r101_c1	**3.7**	30.9	c101_r1	**35.7**	313.5
r101_c2	**3.3**	30.9	c105_c2	**52.9**	**52.9**
r101_rc1	**4.7**	41.3	c105_rc1	**59.5**	302.9
r105_c1	**17.2**	7072.7	c105_r1	**500.8**	522.0
r105_c2	**11.8**	6269.7	c106_c2	81.1	**46.4**
r105_rc1	**11.3**	2128.9	c106_rc1	11.6	**11.2**
r201_c1	**1333.6**	6080.4	c106_r1	**45.8**	191.1
r201_rc1	**81.6**	730.2	c107_c2	**59.2**	62.7
rc101_c1	**4.1**	1425.6	c107_rc1	**367.1**	1320.2
rc101_c2	**3.8**	859.0	c107_r1	**291.4**	10653.4
rc101_r1	**10.6**	3102.6	c201_rc1	935.0	**18.2**
rc201_c1	**215.9**	11185.3	c201_r1	2674.1	**126.2**
rc201_c2	**51.9**	1580.0	c201_c1	419.7	**20.0**
c101_c2	**19.2**	50.1	c205_rc1	**571.2**	13620.7
c101_rc1	**11.6**	14.2	c205_r1	**5898.6**	7181.7
Mean	**118.9**	2706.8			
Closed	**118**	30			

Table D.9: Computational time of the BOTOPTW on the instances $c101$ and $pr01$.

Instances	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}	Instances	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}
c101_1_15	11.9	15.0	**8.3**	9.7	10.7	11.5	pr01_1_15	22.3	16.6	12.2	**11.0**	13.7	16.5
c101_1_20	23.3	16.9	**12.4**	13.6	13.7	17.0	pr01_1_20	68.8	65.8	**24.3**	40.7	31.3	52.6
c101_1_25	30.8	29.3	15.7	**14.9**	18.3	22.7	pr01_1_25	128.2	101.5	**50.6**	70.5	60.8	102.5
c101_1_30	37.4	27.6	**18.9**	22.7	24.3	28.0	pr01_1_30	1468.8	1240.1	**128.0**	405.7	286.5	413.4
c101_1_35	47.7	35.6	**24.3**	**24.3**	28.3	36.5	pr01_1_35	1678.2	952.9	**346.0**	664.9	764.88	834.7
c101_2_15	28.9	26.9	**18.8**	19.0	20.0	22.9	pr01_2_15	25.0	24.9	**15.5**	17.0	21.3	29.1
c101_2_20	56.8	52.6	46.3	41.1	**40.3**	58.4	pr01_2_20	126.9	106.7	89.4	**86.9**	1339.6	142.3
c101_2_25	154.7	118.3	106.1	103.1	**92.8**	115.3	pr01_2_25	2905.7	1942.5	2097.5	1796.1	1784.5	**251.7**
c101_2_30	135.2	147.6	125.1	**94.5**	99.5	112.8	pr01_2_30	757.8	624.5	5423.0	**553.9**	-	674.6
c101_2_35	78.2	93.4	**58.7**	**58.7**	68.0	78.0	pr01_2_35	-	-	-	-	-	-
c101_3_15	25.7	28.9	**20.8**	20.9	22.96	28.6	pr01_3_15	29.8	23.8	19.2	**17.1**	21.6	24.6
c101_3_20	356.1	236.5	249.9	184.8	**152.3**	460.6	pr01_3_20	68.0	64.9	**41.4**	48.0	52.4	65.6
c101_3_25	2712.7	1846.3	1809.1	**1195.7**	1536.3	2993.0	pr01_3_25	202.5	197.6	**112.8**	138.0	145.3	199.8
c101_3_30	-	-	-	-	-	-	pr01_3_30	551.6	515.6	**279.3**	380.7	341.7	504.3
c101_3_35	-	-	-	-	-	-	pr01_3_35	2865.6	**2528.8**	2815.7	2562.4	2604.6	-
c101_4_15	27.7	24.7	**20.4**	22.7	21.92	25.2	pr01_4_15	25.1	24.1	19.3	**16.8**	20.9	28.2
c101_4_20	372.8	176.5	247.1	**139.5**	177.4	458.9	pr01_4_20	67.2	73.1	50.0	47.4	**46.4**	74.7
c101_4_25	4110.8	5224.1	5821.6	3318.6	**2725.9**	4643.3	pr01_4_25	236.6	192.3	**102.4**	133.2	131.0	193.2
c101_4_30	-	-	-	-	-	-	pr01_4_30	472.8	440.5	**260.4**	304.1	278.1	415.6
c101_4_35	-	-	-	-	-	-	pr01_4_35	2530.0	2507.3	2693.6	**2167.8**	2306.3	-
Mean	1196.2	1140.6	1211.4	1029.0	**894.9**	1223.6	Mean	504.8	373.9	228.0	261.1	333.7	**209.3**

Appendix D. Results on the BOTOPTW

This appendix aims to see the impact of each technique of Section 3 in SeM on BOTOPTW instances. Tables D.9 and D.10 provides the computation times in seconds (*Time*) for SeM and SeM improved with each technique. The column called T_{gap} (respectively T_{lb}, T_{valid}, T_{save} and T_{dual}) represents the algorithm SeM improved with the technique T_{gap} (respectively T_{lb}, T_{valid}, T_{save} and T_{dual}). The line *Mean* represents the average computation times for the instances closed by all methods, and the line *Closed* represents the number of closed instances. Figure D.6 represents the performance profiles of SeM and SeM improved with each technique.

Thanks to the mean CPU time and the performance profiles, we can see that all techniques provide improvements to SeM, in particular T_{lb} and T_{valid}.

Table D.10: Computational time of the BOTOPTW on the instances $r101$ and $rc101$.

Instances	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}	Instances	SeM	T_{gap}	T_{lb}	T_{valid}	T_{save}	T_{dual}
r101_1_15	9.0	6.5	**4.5**	5.2	6.7	7.6	rc101_1_15	10.0	7.3	4.5	4.7	**3.7**	8.6
r101_1_20	12.9	9.5	**6.2**	6.5	7.5	9.6	rc101_1_20	10.5	9.9	**6.0**	6.3	6.7	11.4
r101_1_25	16.3	11.7	**7.6**	8.1	9.4	15.0	rc101_1_25	18.1	16.2	8.6	8.6	**7.1**	13.2
r101_1_30	20.3	22.5	**12.6**	12.7	13.3	20.6	rc101_1_30	21.5	15.8	**10.0**	10.4	**10.0**	18.8
r101_1_35	38.4	33.6	**19.8**	22.7	24.1	39.5	rc101_1_35	24.9	19.6	**11.8**	12.2	13.3	18.9
r101_2_15	15.7	15.8	**9.8**	11.7	16.6	15.4	rc101_2_15	19.9	15.8	**9.9**	10.9	14.7	16.1
r101_2_20	23.1	23.4	**14.1**	15.6	21.3	23.7	rc101_2_20	29.2	23.3	**15.1**	15.2	20.2	27.7
r101_2_25	42.0	33.5	**19.5**	21.2	27.5	33.3	rc101_2_25	31.7	36.4	**18.5**	20.1	26.6	31.7
r101_2_30	52.9	60.1	**31.9**	33.1	40.9	52.8	rc101_2_30	40.1	38.1	**23.5**	30.5	31.8	44.7
r101_2_35	74.8	72.6	**41.3**	48.5	65.7	70.7	rc101_2_35	44.1	45.4	**27.5**	30.9	39.3	44.8
r101_3_15	28.3	27.8	**16.6**	19.1	25.7	27.6	rc101_3_15	18.5	15.9	**10.0**	11.0	13.6	15.8
r101_3_20	39.1	36.1	28.5	**25.4**	34.1	37.7	rc101_3_20	30.2	29.3	**18.3**	20.9	26.4	29.0
r101_3_25	58.5	55.8	**32.6**	37.4	43.9	55.3	rc101_3_25	50.8	49.1	**31.1**	34.2	43.5	49.7
r101_3_30	78.4	79.4	**46.9**	55.5	71.6	80.8	rc101_3_30	59.8	59.5	**36.9**	47.9	50.6	58.4
r101_3_35	111.8	131.7	80.9	**78.4**	99.3	113.9	rc101_3_35	70.0	70.2	55.6	**50.0**	60.3	69.6
r101_4_15	32.9	31.5	**18.4**	22.0	30.0	37.0	rc101_4_15	15.6	15.8	**10.0**	12.5	15.0	18.4
r101_4_20	47.3	46.3	34.9	**31.9**	44.5	45.7	rc101_4_20	33.0	31.6	**19.4**	26.0	29.9	31.2
r101_4_25	70.3	68.2	**38.6**	44.7	59.6	80.0	rc101_4_25	54.2	54.1	**34.1**	44.0	49.6	63.1
r101_4_30	120.5	100.9	77.2	**65.6**	85.7	99.7	rc101_4_30	71.0	72.5	**45.0**	51.4	73.0	70.1
r101_4_35	157.6	158.8	**89.7**	108.0	143.3	156.3	rc101_4_35	93.3	92.3	**58.6**	66.0	93.2	91.9
Mean	52.5	51.3	**31.6**	33.7	43.5	51.1	Mean	37.3	35.9	**22.7**	25.7	31.4	36.6

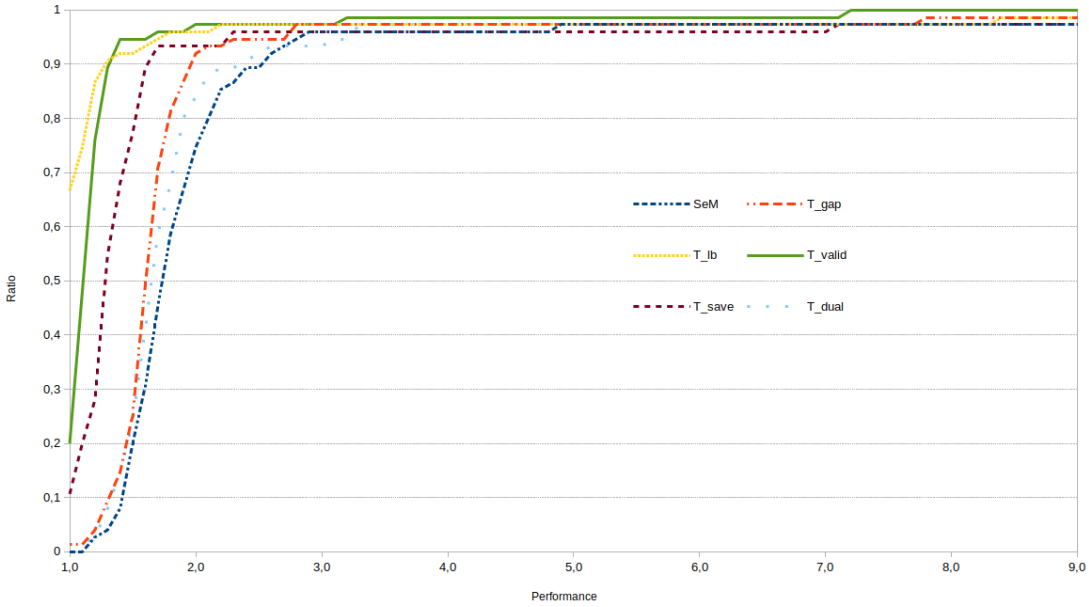


Figure D.6: Performance profiles of SeM and SeM improved with each technique on BOTOPTW instances.