An exact column generation-based algorithm for Bi-Objective Vehicle Routing Problems

Estèle Glize, Nicolas Jozefowiez, Sandra Ulrich Ngueveu

ISCO 2028
An exact column generation-based algorithm for Bi-Objective Vehicle Routing Problems

Estèle Glize[1(✉)], Nicolas Jozefowiez[2], and Sandra Ulrich Ngueveu[1]

[1] CNRS, LAAS, INSA, INP Toulouse, Toulouse, France,
glize,ngueveu@laas.fr,
[2] LCOMS, Université de Lorraine, Metz, France,
nicolas.jozefowiez@univ-lorraine.fr

Abstract. We propose a new exact method for bi-objective vehicle routing problems where edges are associated with two costs. The method generates the minimum complete Pareto front of the problem by combining the scalarization of the objective function and the column generation technique. The aggregated objective allows to apply the exact algorithm for the mono-objective vehicle routing problem of Baldacci et al. (2008). The algorithm is applied to a bi-objective VRP with time-windows. Computational results are compared with a classical bi-objective technique. The results show the pertinence of the new method, especially for clustered instances.

Keywords: Combinatorial MOP, vehicle routing problem, exact method, column generation

1 Introduction

This paper proposes a competitive method for solving to optimality a variant of the *vehicle routing problem* (*VRP*) [1]: the *bi-objective VRP* (*BOVRP*) where edges are associated with two costs. These objectives can be conflictive: in motor vehicle, the travel time differs from the distance. In this case, solving to optimality means finding the non-dominated set. Many industrials are interested in finding a good compromise.

Multi-objective *VRPs* (*MOVRPs*) are more and more studied. A complete survey of *MOVRP* can be found in *Jozefowiez et al.* [2]. In addition to the minimization of travel distance, most *MOVRPs* aim to minimize the number of vehicles or to maximize the fairness of routes.

In the larger scope of multi-objective integer programming (*MOIP*), exact methods are divided into two classes: methods working on the feasible solution space [3] and those working on the objective space [4]. These last methods solve a sequence of mono-objective problems and so, rely on the efficiency of single-objective integer programming solvers. The ϵ-constraint method is the most commonly used objective space search algorithm [5–7] as its efficiency is verified. The

balanced box method of *Boland et al.* [8] shows significant improvements for solution of *MOIPs* by exploring the decision space smartly. Recently, the efficient method of *Dai and Charkhgard* [9] which combines the balanced box method and the ϵ-constraint technique, has been applied to a *2-Dimensional Knapsack Problem* and to the *bi-objective Assignment Problem*.

Section 2 gives preliminaries about bi-objective optimization and an introduction of an algorithm to solve the mono-objective *VRP*. It also defines a formulation of the *BOVRP*. The branch-and-price method is presented in Section 3. Then, Section 4 introduces a classical bi-objective technique and computationally compares the two methods. Finally, Section 5 concludes about this new method.

2 Preliminaries

2.1 Problem Definition

Let $G = (V, E)$ be a non-oriented graph. A node $i \in V \backslash v_0$ is called a customer and has a demand q_i. These demands are satisfied by a fleet of K vehicles of capacity Q. A vehicle k starts and returns at a node v_0 called the depot and performs a route r_k by passing through a set of customers. The route r_k is said to be feasible if the total capacity of a vehicle is not exceeded by the demands.

An edge $e \in E$ of the graph has two costs c_e^1 and c_e^2. Each route r_k provides two costs c_k^1 and c_k^2 representing the sum of the two costs on the used edges. The aim of the studied *BOVRP* is to minimize the sum of each cost of the routes used.

Let Ω be the set of feasible routes r_k and a_{ik} be equal to 1 if the customer i belongs to the route r_k. The Set Partitioning formulation of the *BOVRPR* [10] is stated in the model (1).

$$\begin{cases} minimize \ (\sum\limits_{r_k \in \Omega} c_k^1 \theta_k, \sum\limits_{r_k \in \Omega} c_k^2 \theta_k) \\ \sum\limits_{r_k \in \Omega} a_{ik} \theta_k \geq 1 \ (v_i \in V \backslash \{v_0\}), \\ \sum\limits_{r_k \in \Omega} \theta_k \leq K, \\ \theta_k \in \{0, 1\} \ (r_k \in \Omega). \end{cases} \tag{1}$$

where θ_k is a variable that indicates if the route $r_k \in \Omega$ is selected in the solution ($\theta_k = 1$) or not ($\theta_k = 0$).

2.2 Single objective algorithm for the *VRP*

The method presented in this paper works on the objective space and is based on the exact algorithm for the single objective *VRP* of Baldacci et al. [12].

This state-of-the-art method considers the set partitioning formulation of the *VRP* as a master problem *MP*. As this formulation contains an exponential number of variables θ_k, $r_k \in \Omega$, *MP* needs to be solved optimally on a reduced set of columns. This set is obtained with a three steps algorithm:

1. Compute a good lower bound *LB* by column generation algorithm and a good upper bound *UB*. Compute the gap $\gamma = UB - LB$.
2. Generate all routes with a reduced cost lower than γ. Indeed, it can be proven that routes with a reduced cost higher than γ cannot be in the optimal integer solution. Let $\overline{\Omega}$ be this reduced set of routes.
3. Solve the initial integer problem on $\overline{\Omega}$ to obtain the optimal (integer) solution.

The second step uses dynamic programming to generate the lower bounds on reduced cost necessary to go from the depot to the node i with a load lower than q in non-elementary paths. Then, it solves an *elementary shortest path problem with resource constraints* (*ESPPRC*) with a bi-directional labeling algorithm to produce interesting paths. Finally, feasible routes are produced by combining pairs of these paths.

As previously mentioned, this method will be used to solve the *BOVRP*. In the following, we will refer to the second step by *GENROUTE*(*UB*,*LB*) with *UB* and *LB* the upper bound and the lower bound previously computed.

2.3 Multi-objective optimization

The main purpose of this work is to obtain, in a-posteriori fashion, the minimum complete Pareto front of the *BOVRP*. All concepts of multi-objective optimization are detailed in [11], but an introduction is given in the following of the paper.

Let denote Θ the set of combinations of θ_k, $r_k \in \Omega$, which lead to a feasible solution. An element $\theta \in \Theta$ is a binary vector of size $card(\Omega)$ with $card(\Omega)$ the size of the set Ω indicating which routes are in the solution θ. For $\theta \in \Theta$, let $F(\theta) = (c_1(\theta), c_2(\theta)) = (\sum_{r_k \in \Omega} c_k^1 \theta_k, \sum_{r_k \in \Omega} c_k^2 \theta_k)$ be the function vector to minimize. $\mathcal{Y} = F(\Theta)$ represents the objective space and $y = F(\theta) \in \mathcal{Y}$ a point in the objective space. The following definitions are only valid for a minimization problem and specify the output of the method.

Definition 1. $(a, b) \in \Theta^2$.

$$a \text{ is Pareto dominant with respect to } b \Leftrightarrow \begin{cases} f_i(a) \leq f_i(b) \ \forall i \in \{1,2\} \\ f_i(a) < f_i(b) \ \exists i \in \{1,2\} \end{cases}$$

Definition 2. *A solution $a \in \Theta$ is said to be an efficient (or a Pareto-optimal) solution if $\nexists b \in \Theta$, $b \neq a$, such that b is Pareto dominant with respect to a.*

Definition 3. *A point $y \in \mathcal{Y}$ is said to be a non-dominated point if the solution $a \in \Theta \backslash F(a) = y$ is an efficient solution.*

Definition 4. *A non-dominated point $y \in \mathcal{Y}$ is said to be supported if it is located on the boundary of the convex hull of $\mathcal{Y}$. A non-dominated point $y \in \mathcal{Y}$ is said to be non-supported if it is located on the interior of the convex hull of $\mathcal{Y}$.*

A complete Pareto front is the set of all non-dominated points of the problem. Furthermore, as the same point $y \in \mathcal{Y}$ can be associated with several different solutions in Θ, the number of efficient solutions can be larger than the number of non-dominated points. In this paper, the method provides the minimum Pareto front that-is-to-say that only one efficient solution per non-dominated point is provided.

To lighten the notation, we will refer to the costs of a point $y \in \mathcal{Y}$ associated to a solution $\theta \in \Theta$, by $c^1(y)$ and $c^2(y)$ instead of the cost of the solution $c^1(\theta)$ and $c^2(\theta)$.

3 *Two-step* Method

3.1 Global Algorithm

The *two-step* method is an objective space search method, that is to say that the algorithm determines areas in the objective space in which non-dominated points could be present. Once the areas are delimited, the method scalarizes the objectives to apply a single objective method to go through them. The scalarization we use is a weighted-sum of the two objectives [13]. The set partitioning formulation in this case is Model (2), where all notations are the same for the formulation (1). It is called the master problem MP_λ and its linear relaxation is denoted LMP_λ.

$$\begin{cases} minimize\ \lambda \sum\limits_{r_k \in \Omega} c_k^1 \theta_k + (1-\lambda) \sum\limits_{r_k \in \Omega} c_k^2 \theta_k \\ \sum\limits_{r_k \in \Omega} a_{ik} \theta_k \geq 1\ (v_i \in V \backslash \{v_0\}), \\ \sum\limits_{r_k \in \Omega} \theta_k \leq K, \\ \theta_k \in \{0,1\}\ (r_k \in \Omega). \end{cases} \tag{2}$$

Let introduce the call of the second step of *Baldacci et al.* method for the weight λ: *GENROUTE* (UB, LB, λ). In this algorithm, the gap γ is computed with respect to λ: $\gamma = \lambda(c^1(UB) - c^1(LB)) + (1-\lambda)(c^2(UB) - c^2(LB))$. We also denote $c(S)_\lambda$ as the weighted cost of a point S for the weight λ such that $c(S)_\lambda = \lambda c^1(S) + (1-\lambda)c^2(S)$.

The algorithm of the *two-step* method returns the set Θ of all non-dominated points of the *BOVRP* and is described in Algorithm 1. It is decomposed in two steps as the algorithm of Ulungu and Teghem [13]. First, the supported points are computed thanks to the function *findSupportedPoint* described in Section

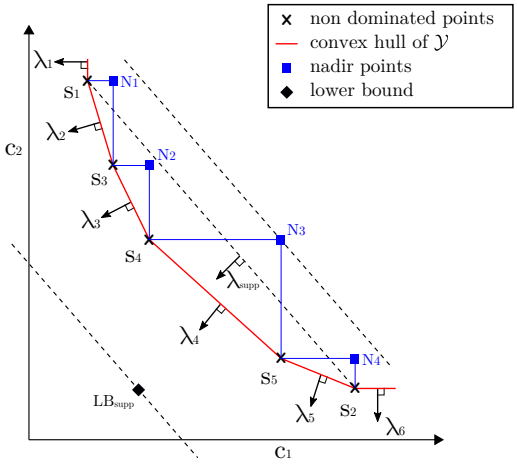

Fig. 1. Example front with supported points and their nadir points.

3.2. Let S_1 and S_2 denote the optimal solutions that minimize the costs c^1 and c^2 respectively. The aim of the first step is to generate all routes that can conduct to a non-dominated point in the triangle defined by S_1, S_2 and their ideal point $I = (c^1(S_1), c^2(S_2))$. These set of routes are returned as $\overline{\Omega_{supp}}$. All supported points are also returned as they are optimal solutions of MP_λ, for some λ [14]. Figure 1 represents a Pareto front with only supported points S_1 to S_5. S_1 minimizes the first cost c_1 ($\lambda_1 = 1$) and S_2 minimizes the second cost c_2 ($\lambda_6 = 0$). For instance, $\forall \lambda \in [\lambda_2; \lambda_3]$, S_3 is the optimal solution of MP_λ.

Then, the non-supported points are found in areas not explored yet. These areas are triangles defined by two consecutive supported points and their nadir point $N = (c^1(S_2), c^2(S_1))$. The search in a triangle is performed by the function *findAllPoint* detailed in Section 3.3.

The intermediate functions *getOptimalSolution* and *gradient* are used in the global algorithm and are described in Algorithm 2 and 3 respectively. The first one takes a direction λ in input and aims to return the optimal solution of MP_λ using the *GENROUTE* algorithm. The other gives the gradient of the line between the two points it receives in input.

3.2 First step

The first step is defined by the function *findSupportedPoint*$(S_1, S_2, \lambda_{supp})$ in Algorithm 4. It needs as input the non-dominated points S_1 and S_2 and a direction λ_{supp}. It generates all routes that can conduct to a non-dominated point situated below the line (S_1S_2).

To do so, we compute the optimal solution LB_{supp} of $LMP_{\lambda_{supp}}$. Then, we apply *GENROUTE* (UB=S_1,LB_{supp},λ_{supp}) to have a reduced set of routes $\overline{\Omega_{supp}}$.

Algorithm 1 *Two-step* method

Input: A graph G representing the $BOVRP$
Output: A set Θ of all non-dominated points

Set $\Theta = \emptyset$;
$S_1 \leftarrow$ *getOptimalSolution*(1); // S_1 the optimal solution minimizing c_1
$S_2 \leftarrow$ *getOptimalSolution*(0); // S_2 the optimal solution minimizing c_2
$\Theta = \Theta \cup \{S_1\} \cup \{S_2\}$;
$\lambda_{supp} \leftarrow$ *gradient*(S_1,S_2);
$(\Theta,\overline{\Omega_{supp}}) \leftarrow$ *findSupportedPoint*(S_1,S_2,λ_{supp}); // return all supported points and a set of routes
Compute $c(S_1)_{\lambda_{supp}} = \lambda_{supp}c^1(S_1) + (1 - \lambda_{supp})c^2(S_2)$;
for S_i and S_j two consecutive points in Θ such that $c^1(S_i) < c^1(S_j)$ **do**
    $\Theta \leftarrow$ *findAllPoint*(S_i,S_j,$c(S_1)_{\lambda_{supp}}$, λ_{supp},$\overline{\Omega_{supp}}$);
end for
return Θ

Algorithm 2 *getOptimalSolution*(λ)

Input: A value λ
Output: The optimal solution S of MP_λ

Solve LMP_λ to obtain a lower bound LB;
Solve MP_λ to obtain an upper bound UB;
$\overline{\Omega} \leftarrow GENROUTE(UB, LB, \lambda)$;
Solve MP_λ on $\overline{\Omega}$ to obtain the optimal solution S;
return S;

Algorithm 3 *gradient*(S_i,S_j)

Input: Two points S_i and S_j
Output: The gradient of (S_iS_j)

$\lambda = \frac{abs(c^1(S_j) - c^1(S_i))}{abs(c^2(S_i) - c^1(S_i) - c^2(S_j) + c^1(S_j))}$;
return λ;

By taking the point S_1 as upper bound and the direction λ_{supp}, we ensure that all non-dominated points situated below the line $(S_1\ S_2)$ is represented by a combination of routes in $\overline{\Omega_{supp}}$

Algorithm 4 *findSupportedPoint*(S_1,S_2,λ_{supp})

Input: Points S_1 and S_2 and the direction λ_{supp}
Output: Set Θ_{supp} of supported points. Set of routes $\overline{\Omega_{supp}}$

Set $\Theta_{supp} = \emptyset$ and $\overline{\Omega_{supp}} = \emptyset$;
Solve $LMP_{\lambda_{supp}}$ to obtain LB_{supp};
$\overline{\Omega_{supp}} \leftarrow$ *GENROUTE*(UB=S_1,LB_{supp},λ_{supp});
dichotomicSearch(Θ_{supp},S_1,S_2,$\overline{\Omega_{supp}}$);
return (Θ_{supp},$\overline{\Omega_{supp}}$);

Algorithm 5 *dichotomicSearch*(Θ_{supp},S_i,S_j,$\overline{\Omega_{supp}}$)

Input: Set Θ_{supp}. 2 points S_i and S_j. Set of routes $\overline{\Omega_{supp}}$.

$\lambda_i \leftarrow$ *gradient*(S_i,S_j);
Solve MP_{λ_i} on $\overline{\Omega_{supp}}$ to obtain the optimal solution S_k;
if $S_k \neq S_i$ and $S_k \neq S_j$ **then**
  $\Theta_{supp} = \Theta_{supp} \cup \{S_k\}$;
  if $c^1(S_i) + 1 < c^1(S_k)$ and $c^2(S_i) - 1 > c^2(S_k)$ **then**
    dichotomicSearch(Θ_{supp},S_i,S_k,$\overline{\Omega_{supp}}$);
  end if
  if $c^2(S_j) + 1 < c^2(S_k)$ and $c^1(S_k) - 1 > c^1(S_k)$ **then**
    dichotomicSearch(Θ_{supp},S_k,S_j,$\overline{\Omega_{supp}}$);
  end if
end if

Finally, we search all supported points in $\overline{\Omega_{supp}}$ in a dichotomic approach summarized in Algorithm 5.

After the first step, all supported non-dominated points are already found. It calls the algorithm *GENROUTE* only once and solves the integer problem for each supported points. However, the set of routes $\overline{\Omega_{supp}}$ contains important information and has to be returned to be used in the second step.

3.3 Second step

The second step aims to explore each upper right triangle defined by two consecutive non-dominated points computed in the first step and their nadir point. In practice, input data are integers, therefore we substract one to each coordinate of the nadir because non-dominated points having a same coordinate than the nadir are weakly dominated. For instance, in Figure 1, if a point is

Algorithm 6 *findAllPoint*(S_i,S_j,$c(S_1)_{\lambda_{supp}}$,λ_{supp},$\overline{\Omega_{supp}}$)

Input: Points S_i and S_j. The cost $c(S_1)_{\lambda_{supp}}$, the direction λ_{supp} and set $\overline{\Omega_{supp}}$ of the first step
Output: The set Θ_{nd} of non-dominated points

```
Set N = (c^1(S_j) − 1, c^2(S_i) − 1); Set Θ_nd = ∅
if c(N)_{λ_supp} ≤ c(S_1)_{λ_supp} then
   findInTriangle(Θ_nd,S_i,S_j,Ω_supp);
else
   λ_i ← Gradient(S_i,S_j);
   Solve LMP_{λ_i} to obtain LB_i;
   Ω_i ← GENROUTE(N,LB_i,λ_i);
   findInTriangle(Θ_nd,S_i,S_j,Ω_i)
end if
return Θ_nd;
```

located on the segment $[N_1S_3]$ (resp. $[S_1N_1]$), it is strictly dominated by S_3 (resp. S_1). The search is performed as explained in Algorithm 6. It requires in input two non-dominated points S_i and S_j such that $c^1(S_i) < c^1(S_j)$. First, a condition has to be checked: if the nadir $N = (c^1(S_j)-1, c^2(S_i)-1)$ is such that $c(N)_{\lambda_{supp}} \leq c(S_1)_{\lambda_{supp}}$, then directly apply the function *findInTriangle* to search on the set of known routes $\overline{\Omega_{supp}}$. Indeed, if $c(N)_{\lambda_1} \leq c(S_1)_{\lambda_1}$, we already have generated all routes that can conduct to a non-dominated point in this triangle in the first step. Otherwise, some routes has to be generated before applying *findInTriangle* because the nadir point is upside the line (S_1S_2).

Figure 1 represents a partial front obtained after the first step. The second step explores the triangles defined by two consecutive points S and their nadir N like $S_3S_4N_2$ or $S_4S_5N_3$. We already have generated in the first phase all routes that can conduct to a non-dominated point below the dotted line passing through S_1 and S_2. So, all interesting routes for non-dominated points in $S_3S_4N_2$ are already generated because the nadir N_2 is under this line. The condition $c(N_2)_{\lambda_{supp}} \leq c(S_1)_{\lambda_{supp}}$ is satisfied. On the contrary, we don't have all the routes that can conduct to a non-dominated point in $S_4S_5N_3$ as the dotted line passing through N_3 is above the dotted line passing through S_1 and S_2. So, we have the condition $c(N_3)_{\lambda_{supp}} > c(S_1)_{\lambda_{supp}}$.

The algorithm *findInTriangle*(Θ_{nd},S_i,S_j,$\overline{\Omega}$) is similar to *DichotomicSearch*. It works on the computed set $\overline{\Omega}$ given in input and requires two non-dominated points S_i and S_j and their gradient λ_i. It solves the integer problem MP_{λ_i} on $\overline{\Omega}$ with two additional constraints: $\sum_{r_k \in \Omega} c_k^1 \theta_k \leq c^1(N)$ and $\sum_{r_k \in \Omega} c_k^2 \theta_k \leq c^2(N)$. If there is no optimal solution, it means that there is no non-dominated points in the area and the function stops. Otherwise, if there is an optimal solution S_k, S_k is added to Θ_{nd} and the function *findInTriangle* is called again for (S_i,S_k) and (S_k,S_j).

4 Computational Experiments

To compare proposed the *two-step* method to the state-of-the-art, we have implemented a *reference* method. It is the more direct way to use the *Baldacci et al.* method in an ϵ-constraint technique as it is classically done in the literature. The algorithm, summarized in Algorithm 7, is based on the ϵ-constraint formulation that aims to minimize the first cost c^1 under the constraint that the second cost has to be lower than a certain value ϵ.

At the beginning, ϵ is set to $+\infty$. At each iteration, the reference method consists in finding a lower bound LB and an upper bound UB. Then, it applies the mono-objective algorithm $GENROUTE(UB,LB)$ to obtain $\overline{\Omega}$, the restricted set of routes with (i) their reduced cost within the gap $\gamma = UB - LB$ and (ii) below the constraints that the second cost is lower than ϵ. The algorithm optimally solves the integer problem restricted to $\overline{\Omega}$. If a new integer solution is found, ϵ is set to the value of the second objective of the solution minus one and the process is repeated. If no new optimal solution is found, the algorithm stops.

Algorithm 7 Algorithm of the reference method

$\epsilon \leftarrow +\infty$
while $\exists$ a solution **do**
  Solve the linear relaxation of the problem for ϵ to obtain LB
  Find a feasible solution UB of the integer problem for ϵ
  $\overline{\Omega} \leftarrow GENROUTE(UB, LB)$
  Solve the integer problem on $\overline{\Omega}$ to obtain S_{OPT}
  if $\exists S_{OPT}$ **then**
    Set $\epsilon \leftarrow {S_{OPT}}^2 - 1$
  end if
end while

Results. To the best of our knowledge, there is no benchmark for multi-objective vehicle routing problems with different costs on edges. Therefore we propose new instances for the BOVRP with time windows ($BOVRPTW$) which minimizes two different route costs. Each instance is a combination of two Solomon's instances. The first instance provides the first edge costs, the time windows, the charges and the capacities. The second instance only provides the second edge costs. We have tested the algorithm on 20 instances of 25 customers and 20 instances of 50 customers, which correspond to the 25 and 50 first customers of Solomon's instances. Each method returns the minimum complete Pareto Front of the $BOVRPTW$.

The experiments have been conducted on a Xeon E5-2695 processor with a 2.30GHz CPU and 3.5Go in a single thread. The implementation is in C++ and the linear problems and the integer problems are solved with Gurobi 7.1. The

time limit for all experiments is 6 hours. Results are reported in Table 1 for the 27 instances for which at least one of the method tested converged.

Table 1 presents the features of the instances like the number of customers, if it is a clustered instance or not, the number of strict non-dominated points and the number of non-supported non-dominated points in the final Pareto front. It also shows the mean CPU time in seconds on 10 executions as well as their standard deviations for each method. We can notice that the *reference* method dominates the *two-step* method on instances easily solved - inferior to 49 seconds for the two methods. On the contrary, the *two-step* method dominates the other on harder instances. Furthermore, the instances noted 9, 21, 22 and 23 of 25 customers are only solved by the *two-step* method. 3 of them are clustered instances with very few non-dominated points in the final exact Pareto front. It suggests that the *two-step* method outperforms the state-of-the-art method for graphs with clustered structure.

Table 1 also exhibits the execution time in seconds for the methods in graphs with 50 customers (instances from 24 to 30). The number of non-dominated points is, as expected, larger than for the graphs with 25 clients. The execution of the algorithms is more time consuming and only converges for 2 instances in the *reference* method and for 7 in the *two-step* methods over 20 tested instances.

5 Conclusion

In this paper, we proposed an exact method to solve the bi-objective vehicle routing problem : the *two-step* method. It scalarizes the objective function and uses column generation to find all non-dominated points, supported and non-supported points. The method is also generic for all classes of *BOVRP* as it doesn't exploit any specific property.

To show the efficiency of the scheme proposed we have implemented the ϵ-constraint method combined with the *GENROUTE* component. Computational experiments showed that the *two-step* method outperforms this reference method, especially for clustered graphs.

References

1. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science 6, 80–91 (1959)
2. Jozefowiez, N., Semet, F., Talbi, E.-G.: Multi-objective vehicle routing problems. European journal of operations research 189, 293–309 (2008)
3. Parragh, S.N., Tricoire, F.: Branch-and-bound for bi-objective integer programming. Optimization online (2015)
4. Boland, N., Charkhgard, H., Savelsbergh, M.: The triangle splitting method for bi-objective mixed integer programming. International Conference on Integer Programming and Combinatorial Optimization, 162–173 (2014)
5. Moradi, S. , Raith, A., Ehrgott, M.: A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem. European Journal of Operational Research 244, 369–378 (2015)

	Instance Features					*Reference* method		The *two-step* method	
#	Instance	Number of customers	Clustered?	Number of non-dominated points	Number of non-supported points	mean time (s)	σ (s)	mean time (s)	σ (s)
1	R105_C1	25	no	33	28	**31**	1	32	1
2	R105_C2	25	no	32	25	**24**	1	49	2
3	R109_C1	25	no	26	14	1055	130	**289**	30
4	R109_C2	25	no	28	17	1028	120	**464**	41
5	R102_RC1	25	no	68	55	-	-	**2113**	152
6	R105_RC1	25	no	34	28	**24**	1	44	2
7	R109_RC1	25	no	45	34	1794	187	**660**	51
8	RC101_R1	25	no	27	16	**18**	1	42	1
9	RC105_R1	25	no	19	13	193	19	**139**	10
10	RC106_R1	25	no	23	14	445	57	**253**	42
11	RC101_C1	25	no	9	3	8	0	8	0
12	RC102_C1	25	no	18	12	3823	382	**473**	37
13	RC105_C1	25	no	21	14	191	14	**49**	2
14	RC105_C2	25	no	22	13	204	16	**38**	2
15	RC106_C1	25	no	14	8	233	18	**121**	12
16	RC106_C2	25	no	28	19	456	54	**198**	10
17	C101_C2	25	yes	5	1	10059	782	**6321**	856
18	C106_R1	25	yes	22	14	-	-	**16108**	2227
19	C106_RC1	25	yes	6	4	-	-	**3850**	488
20	C106_C2	25	yes	5	1	-	-	**9389**	1403
21	R101_RC2	50	no	176	155	762	96	**424**	28
22	R105_RC2	50	no	139	118	-	-	**11654**	1220
23	R101_C2	50	no	114	93	476	57	**301**	21
24	R105_C1	50	no	154	134	-	-	**12278**	1226
25	R105_C2	50	no	169	145	-	-	**11897**	1340
26	RC101_R1	50	no	132	116	3651	370	**1982**	65
27	RC105_C2	50	no	79	58	-	-	**17519**	2105

Table 1. Execution time of the methods for some graphs

6. Bérubé, J.-F., Gendreau, M., Potvin, J.-Y.: An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. European Journal of Operational Research 194, 39–50 (2009)
7. Özlen, M., Azizoğlu, M.: Multi-objective integer programming: a general approach for generating all non-dominated solutions. European Journal of Operational Research 199, 25–35 (2009)
8. Boland, N., Charkhgard, H., Savelsberg, M.: A criterion space search algorithm for biobjective integer programming: The balanced box method. INFORMS Journal on Computing 27, 735–754 (2015)
9. Dai, R., Charkhgard, H.: A two-stage approach for bi-objective integer linear programming. Operations Research Letters 46, 81–87 (2018)
10. Balinski, M.L., Quandt, R.E.: On an integer program for a delivery problem. Operations Research 12, 300–304 (1964)
11. Ehrgott, M.: Multicriteria optimization. Springer Science & Business Media (2006)
12. Baldacci, R., Christofides, N., Mingozzi, A.: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 115, 351–385 (2008)
13. Ulungu, E.L., Teghem, J., Fortemps, P.H., Van Nieuwenhuyze, K.: The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. Foundations of Computing and Decision Sciences 20, 149–165 (1995)
14. Geoffrion, A. M.: Proper efficiency and the theory of vector maximization. Journal of Mathematical Analysis and Applications 22, 618–630 (1968)