Column generation algorithms for bi-objective combinatorial optimization problems with a min–max objective

Christian Artigues, Nicolas Jozefowiez, Boadu Sarpong

Noname manuscript No.
(will be inserted by the editor)

Column Generation Algorithms for Bi-Objective Combinatorial Optimization Problems with a Min-Max Objective

Christian Artigues · Nicolas Jozefowiez · Boadu M. Sarpong

Received: date / Accepted: date

Abstract Many practical combinatorial optimization problems can be described by integer linear programs having an exponential number of variables and they are efficiently solved by column generation algorithms. For these problems, column generation is used to compute good dual bounds that can be incorporated in branch-and-price algorithms. Recent research has concentrated on describing lower and upper bounds of bi-objective and general multi-objective problems with sets of points (bound sets). An important issue to address when computing a bound set by column generation is how to efficiently search for columns corresponding to each point of the bound set. In this work, we propose a generalized column generation scheme to compute bound sets for bi-objective combinatorial optimization problems. We present specific implementations of the generalized scheme for the case where one objective is a min-max function by using a variant of the ε-constraint method to efficiently model these problems. The proposed strategies are applied to a bi-objective extension of the multi-vehicle covering tour problem and their relative performances based on different criteria are compared. The results show that good bound sets can be obtained in reasonable times if columns are efficiently managed. The variant of the ε-constraint presented is also better than a standard ε-constraint method in terms of the quality of the bound sets.

Keywords column generation · multi-objective optimization · bound sets · vehicle routing problems

C. Artigues, B. M. Sarpong
LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
Nicolas Jozefowiez
LCOMS, Université de Lorraine, Metz, France
E-mail: nicolas.jozefowiez@univ-lorraine.fr

1 Introduction

The success of exact methods for solving difficult single objective optimization problems relies greatly on the computation and use of lower and upper bounds. For this reason, we can expect that lower and upper bounds are equally important in designing exact methods for multi-objective optimization problems. The number of exact methods for multi-objective optimization problems is relatively small when compared to that of single objective problems and a possible reason is that the idea of bounds for multi-objective problems and efficient methods to compute them are not so well developed. Column generation is one of the most popular methods for computing strong lower bounds for single objective problems that can be modeled by integer linear programs having an exponential number of variables. Nevertheless, column generation is rarely used in the multi-objective case. In this work, we study the design of efficient column generation algorithms for computing strong bounds of bi-objective combinatorial optimization (BOCO) problems by concentrating on the case in which one objective is a min-max function. We will refer to these problems as Bi-Objective Combinatorial Optimization Problems with a Min-Max Objective (BOCOMMO).

A BOCOMMO can be defined by means of a Dantzig-Wolfe decomposition as the selection of a set of columns with minimum total "cost" such that the maximum value of an attribute associated with the set is minimized. More specifically, we consider bi-objective covering problems of the form:

$$\text{Minimize} \quad \sum_{k \in \Omega} c_k \theta_k \tag{1}$$

$$\text{Minimize} \quad \Gamma_{\max} \tag{2}$$

$$\text{subject to:} \quad \sum_{k \in \Omega} a_{ik} \theta_k \geq b_i \qquad (i \in I)\,, \tag{3}$$

$$\Gamma_{\max} \geq \sigma_k \theta_k \qquad (k \in \Omega)\,, \tag{4}$$

$$\theta_k \in \{0,1\} \qquad (k \in \Omega)\,, \tag{5}$$

where θ_k and $\Gamma_{\max}$ are decision variables, Ω is the set of all feasible columns whose description depends on the particular problem, and I is the index set of the covered objects. For each column $k \in \Omega$, c_k and σ_k are two associated costs. The values σ_k are supposed to be integer. That way, we know that it is always possible to set a bound on the second objective in an ϵ-constraint method. However, in some cases, it could be possible for the σ_k to not be integer if the number of possible values is finite or if a step could be computed to ensure that no solution is lost in an ϵ-constraint method. We need to select columns with minimum sum of c_k such that $\Gamma_{\max} = \max_{k \in \Omega}\{\sigma_k \theta_k\}$ is also minimized. Bi-objective generalizations of several combinatorial optimization problems satisfying this condition can be defined. For vehicle routing problems, we generally want to minimize the combined cost of a set of routes such that the value of a property associated with the selected routes (eg. the maximum length

of a route, max capacity of a route, etc.) is also minimized. Another example of a BOCOMMO is a bi-objective extension of the Bin Packing Problem in which we seek to minimize both the number of bins used and the maximum size of a bin.

Multi-Objective Combinatorial Optimization.

The class of problems we will be dealing with (BOCOMMO) is a special case of multi-objective combinatorial optimization (MOCO) problems. A general MOCO problem consists in minimizing a vector of two or more objective functions $F(x) = (f_1(x), f_2(x), \ldots, f_r(x))$ over a finite domain of feasible solutions $\mathcal{X}$. The vector $x = (x_1, x_2, \ldots, x_n)$ is the decision variable, $\mathcal{Y} = F(\mathcal{X})$ corresponds to the images of the feasible solutions in the objective space, and $y = (y_1, y_2, \ldots, y_r)$, where $y_i = f_i(x)$, is a point in the objective space. We say that a solution x' *dominates* another solution x'' if for any index $i \in \{1, 2, \ldots, n\}$, $f_i(x') \leq f_i(x'')$ and there is at least one index $i \in \{1, 2, \ldots, n\}$ for which $f_i(x') < f_i(x'')$. A feasible solution dominated by no other feasible solution is said to be *efficient* or *Pareto optimal* and its image in the objective space is said to be *nondominated*. The set of all efficient solutions is called the *efficient set* (denoted $\mathcal{X}_E$) and the set of all nondominated points is the *nondominated set* (denoted $\mathcal{Y}_N$). In general, more than one efficient solution may correspond to the same nondominated point and so solving a MOCO problem usually means finding at least one efficient solution for each nondominated point. The nondominated set defines what is known as the *Pareto frontier*. An efficient solution that maps onto a nondominated point lying on the convex part of the Pareto frontier is called a *supported efficient solution* and its image is called a *supported nondominated point*.

Lower and Upper Bounds for a MOCO Problem.

Ideal and nadir points are well known lower and upper bounds, respectively, of the non-dominated set $\mathcal{Y}_N$ of a MOCO problem. The coordinates of the ideal point are obtained by optimizing each objective function independently of the others, whereas the coordinates of the nadir point correspond to the worst value of each objective function when we consider the efficient set $\mathcal{X}_E$. Ideal and nadir points are usually poor bounds since they give very little information on where the members of the nondominated set lie (see Figure 1). Given that the solution of a multi-objective problem is a set of solutions rather than a single solution, a better way of defining lower and upper bounds is to use sets of points. [26] were the first to propose the use of sets of points to define bounds for multi-objective problems. They defined a lower bound for a multi-objective integer program as a set of points $\mathbf{L}$ such that the image of each feasible solution of the problem is dominated by at least one of the members of $\mathbf{L}$. A member of $\mathbf{L}$ may or may not correspond to the image of a feasible solution. In a similar way, an upper bound may be defined as a set of points $\mathbf{U}$ corresponding to images of feasible solutions that do not dominate one another.

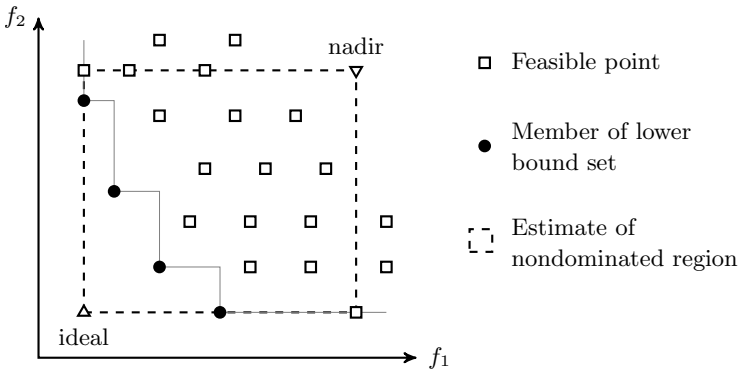


Fig. 1: Lower and Upper Bounds of a Bi-Objective Integer Program.

Recently, the idea of using sets to define bounds for MOCO problems has been used by other authors like [7], [25] and [5]. [7] introduced the terminology *bound sets* to describe the use of sets in defining bounds of a multi-objective problem. They proved some general results and discussed how bound sets can be constructed for bi-objective combinatorial optimization (BOCO) problems by using a weighted sum method [10]. [25] and [5] use ideas similar to those introduced by [7] to compute bound sets for the bi-objective spanning tree problem and the bi-objective binary knapsack problem, respectively.

The main idea used by all these authors in defining a lower bound set is to transform the bi-objective problem into single objective by using a weighted sum method and solve the transformed problem for different weights in order to compute the complete set of supported nondominated points. A lower bound set for the considered bi-objective problem is then defined as the line connecting the set of supported nondominated points. This procedure is only possible if there is an efficient algorithm for solving the single objective problem obtained after the transformation. If the transformed problem is $\mathcal{NP}$-hard, then the set of supported nondominated points for a relaxation of the single objective problem rather needs to be computed (see Figure 1). As pointed out by [7], any scalarization method may be used when computing bound sets but the weighted sum method is mostly used. In spite of the many advantages of using a weighted sum method, it cannot find nonsupported solutions no matter the choice of weights used [4]. If there is a large number of nonsupported points, a lower bound set based on the weighted sum method can be very poor. A first example of using an ε-constraint method [14] in computing bound sets was given by [23]. A disadvantage of an ε-constraint method is that it is not very easy to define values for the parameter ε in such a way that all members of the nondominated set are found. Nevertheless, this is possible for many practical problems including those presented by [17] and [1].

Column Generation in Multi-objective Optimization.

In spite of the importance of column generation and multi-objective optimization, very little has been done in terms of applying column generation to multi-objective optimization problems. [8] treat the use of column generation in integer programming with applications in multi-objective optimization. Their proposed approach consists in first converting a multi-objective problem into single objective through an ε-constraint method before combining column generation and cutting planes to solve the resulting problem. After finding an efficient solution together with the necessary dual information, a form of sensitivity analysis is used to search for neighbouring efficient solutions without changing the value of ε. An example of applying column generation to multi-objective problems has also been presented by [22]. The example uses column generation to create radiotherapy treatment plans. A bi-objective problem that arises in this field is to treat defective cells and also reduce the side effects resulting from the use of radio waves. A continuous convex model in which each column represents a certain treatment dose is proposed for the problem. A weekly plan (feasible point) consists of one or several doses (columns) and a complete treatment program (upper bound set) consists of several weekly treatment plans. The bi-objective problem is first converted into single objective through a weighted sum approach before applying column generation to obtain an approximation to the set of efficient solutions. A similar column generation approach was proposed by [19].

Contributions and Organization of Work.

It is a well known fact that finding the complete nondominated set of a MOCO problem is a difficult and time-consuming task. For this reason, most works that deal with column generation for MOCO problems such as those cited above are mainly interested in finding a set of feasible solutions that do not dominate one another (an upper bound set). In this work, however, we are interested in the design of efficient column generation algorithms for computing strong lower bound sets for BOCO problems in reasonable time. In particular, we propose a generalized column generation scheme for computing lower bound sets of BOCO problems and also present three different strategies for implementing the generalized scheme in the case of BOCO problems with a min-max objective. The proposed strategies are used in computing bound sets for a bi-objective extension of the multi-vehicle covering tour problem and their performances based on different criteria are evaluated. This study is expected to serve a broad range of applications since an important class of combinatorial optimization problems can be formulated as integer programs having an exponential number of variables and these problems are efficiently solved by column generation based algorithms. Moreover, being able to quickly compute good bound sets is expected to contribute to the design of more exact methods for BOCO problems.

In Section 2, we discuss how column generation can be used to compute lower bound sets for a BOCO problem and present implementations for the case of BOCOMMO. An application problem is presented in Section 3 and evaluation of the bound sets computed for this problem as well as comparison of the different column generation strategies are given in Section 4. We end our discussion in Section 5 with concluding remarks and give ideas on how this work can be extended.

2 Column Generation for a BOCOMMO

In order to compute a lower bound set for a BOCOMMO, we need to first transform the BOCOMMO into a single objective problem before solving the single objective problem by column generation. Any scalarization method may be used to transform the BOCOMMO problem but in this work, we will use an ε-constraint problem. A disadvantage of directly adding constraints of the form $\Gamma_{\max} \leq \varepsilon$ to Formulation (1–5) is that solving linear relaxations of the resulting problem can significantly weaken the lower bound set. In order not to unnecessarily weaken the lower bound set, we use a different variant of the standard ε-constraint method which does not require us to directly add constraints of the form $\Gamma_{\max} \leq \varepsilon$ to the formulation. A close examination of Formulation (1–5) reveals that we can decompose a BOCOMMO into two problems since for any set of feasible columns, we can compute the corresponding value of $\Gamma_{\max}$. So, instead of explicitly adding a constraint of the form $\Gamma_{\max} \leq \varepsilon$ to the formulation, we rather drop Constraints 4 and use it to redefine the feasibility of a column. We define a new set of feasible columns $\overline{\Omega^{\varepsilon}}$ where the feasibility of a column k in $\overline{\Omega^{\varepsilon}}$ depends on its associated value σ_k, i.e. $\sigma_k \leq \varepsilon$. Depending on whether we can associate an original column k in Ω can belong to several sets $\overline{\Omega^{\varepsilon}}$. Indeed, the column k can appear in the sets for all the values of ε greater of equal to σ_k. Also, we no longer need to keep the objective function (2) after dropping Constraints (4). In this way, we do not degrade the quality of the linear relaxation. Moreover, if an efficient column generation algorithm exists for the single-objective problem linked with the first objective, we can easily adapt it for the bi-objective problem. We obtain the following problem for a value of ϵ :

$$\text{Minimize} \quad \sum_{k \in \overline{\Omega^{\varepsilon}}} c_k \theta_k \tag{6}$$

$$\text{subject to:} \quad \sum_{k \in \overline{\Omega^{\varepsilon}}} a_{ik} \theta_k \geq b_i \qquad (i \in I)\,, \tag{7}$$

$$\theta_k \in \{0,1\} \qquad (k \in \overline{\Omega^{\varepsilon}})\,. \tag{8}$$

2.1 Computing Lower Bound Sets

For a BOCOMMO, as we suppose the value σ_k to be integer and we can suppose the second objective to be bounded, $\Gamma_{\max}$ can only take on a finite number of values defining a set Γ. If the complete set of feasible columns $\overline{\Omega} = \bigcup_{\varepsilon \in \Gamma} \overline{\Omega^{\varepsilon}}$ is known, a lower bound set can be computed by using a variant of the ε-constraint approach as given in Algorithm 1 in which we consider the following problem :

$$\text{Minimize} \quad \sum_{k \in \overline{\Omega}} c_k \theta_k \tag{9}$$

$$\text{subject to:} \quad \sum_{k \in \overline{\Omega}} a_{ik} \theta_k \geq b_i \qquad (i \in I)\,, \tag{10}$$

$$\theta_k \in \{0,1\} \qquad (k \in \overline{\Omega})\,. \tag{11}$$

The algorithm starts with no restriction on the value of $\Gamma_{\max}$ (that is, $\varepsilon = +\infty$). At each iteration, the linear relaxation of the problem is solved and the optimum as well as the value of $\Gamma_{\max}$ are determined. In the next iteration, the problem is updated to exclude columns k for which σ_k is strictly greater than $\Gamma_{\max}$ (this can be done by removing the columns or setting them to 0). This iterative process continues until the problem becomes infeasible after a finite number of iterations.

Algorithm 1 Computing a Lower Bound Set

1: Set $\mathbf{L} \leftarrow \emptyset$.
2: **while** the problem defined by (9)-(11) is feasible **do**
3: Solve problem defined by (9)-(11) and let c^* and θ^* be the optimum and the optimal solution vector, respectively.
4: Compute $\Gamma_{\max} = \max_{k \in \overline{\Omega}} \{\sigma_k \theta_k^*\}$.
5: $\mathbf{L} \leftarrow \mathbf{L} \cup \{(c^*, \Gamma_{\max})\}$.
6: Set $\theta_k \leftarrow 0$ for all $k \in \overline{\Omega}$ such that $\sigma_k \geq \Gamma_{\max}$.
7: **end while**

In practice, the cardinality of $\overline{\Omega}$ is too large and so a column generation method needs to be used by considering a subset $\overline{\Omega}_1$ of $\overline{\Omega}$. In column generation terminology, the original problem based on the set $\overline{\Omega}$ with an exponential number of columns is referred to as the *master problem* (MP). The MP of a BOCOMMO is given by Formulation (6–8). The linear relaxation of MP which is obtained by writing Constraints (8) as $\theta_k \geq 0$ for $k \in \overline{\Omega}$ is denoted by LMP. If the MP is restricted to a subset $\overline{\Omega}_1$ of $\overline{\Omega}$ we obtain a *restricted master problem* (RMP) whose linear relaxation is denoted by LRMP. By associating Constraints (7) with dual variables π_i $(i \in I)$, the dual formulation of LMP

(DLMP) can be defined as:

$$\begin{aligned} & \text{Maximize} && \sum_{i \in I} b_i \pi_i \\ \text{subject to:} \quad & \sum_{i \in I} a_{ik} \pi_i \leq c_k && (k \in \overline{\Omega}), \\ & \pi_i \geq 0 && (i \in I). \end{aligned}$$

The dual formulation of LRMP (DLRMP) can equally be defined by changing $\overline{\Omega}$ in the above formulation to $\overline{\Omega}_1$.

We note that the set of feasible solutions of LRMP is a subset of the set of feasible solutions of LMP since LRMP is obtained by removing some columns from MP. Removing some columns from LMP, however, implies the removal of some constraints from DLMP and so the set of feasible solutions of DLMP is a subset of the set of feasible solutions of DLRMP. For these reasons, every feasible solution of the LRMP is also a feasible solution of LMP but DLRMP may not always be feasible. If both LRMP and DLRMP are feasible, then the optimal value of LRMP is also the optimal value of LMP and hence a valid lower bound on the optimal value of MP. If LRMP is feasible but DLRMP is infeasible, then some constraints of DLMP are violated by the obtained solution and we need to add one or more of such constraints to DLRMP. Adding constraints to DLRMP corresponds to introducing some new columns to LRMP. In order to know which constraints of DLMP are violated by the DLRMP solution, we need to solve an auxiliary problem referred to as the *subproblem* in column generation terminology. In order to satisfy the constraint $\Gamma_{\max} \leq \varepsilon$ when solving a BOCOMMO by column generation and ε-constraint method, we need to solve a subproblem given by:

$$S(\varepsilon) = \min_{k \in \overline{\Omega} \setminus \overline{\Omega}_1} \left\{ c_k - \sum_{i \in I} \pi_i a_{ik} : \sigma_k \leq \varepsilon \right\}.$$

If the optimal value of the subproblem is negative, then a dual constraint is violated and we need to add the column corresponding to this constraint.

2.2 A Generalized Column Generation Scheme

We propose the generalized column generation scheme in Figure 2 for computing a lower bound set of a BOCO problem. The scheme does not depend on a specific scalarization method and it can be applied to any BOCO problem whose formulation uses an exponential number of column variables. The algorithm starts by first transforming the BOCO problem into single objective by using a scalarization method like a weighted sum or an ε-constraint method. Note that each scalarization method uses a different parameter in transforming a bi-objective problem. For example, a weighted sum method uses a vector of weights to combine the objective functions whereas an ε-constraint method uses a real number ε to restrict one of the objective functions. After transforming

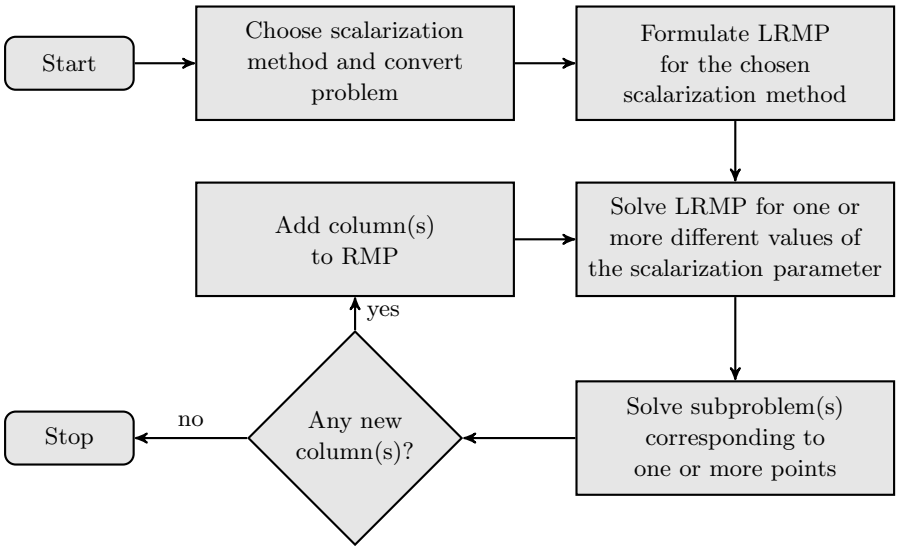


Fig. 2: A Generalized Column Generation Algorithm for BOCO Problems.

the BOCO problem, an LRMP based on the obtained single objective problem is formulated before starting the column generation iterations. An iteration consists in first solving the LRMP (without generating any columns) for one or more different values of the parameter used to transform the BOCO problem. For a weighted sum method, this means varying the value of the vector of weights whereas for an ε-constraint method we have to vary the value of the parameter ε which is used in constraining one of the objectives. After solving the LRMP, we obtain a vector of optimal dual values and a corresponding subproblem for each value of the parameter. Next, we need to solve the subproblem for one or more vectors of dual values. Any relevant columns found are added to the LRMP and the process repeats. If the LRMP is proven to have converged for any value of the parameter, the corresponding point (optimal value of LRMP together with value of the parameter) is saved. The algorithm terminates when the LRMP converges for all relevant values of the parameter. In the case of a weighted sum method, the relevant values are those that are necessary to define the complete set of supported nondominated points of LRMP. For an ε-constraint method, we need to ensure that the image of any feasible point of LRMP is dominated by at least one of the generated points.

Clearly, the exact implementation details of the generalized column generation scheme depends on the scalarization method used but similar subproblems are obtained for all values of the parameter once a scalarization method is chosen. This means that, it is possible to treat more than one subproblem at the same time when searching for relevant columns. For example, it may be possible to easily modify a column found by solving the subproblem for a specific value of the parameter in order to find another column for a different

value of the parameter without having to solve the subproblem for this new value. For several problems, such as Vehicle Routing Problems, solving the subproblem is the most time consuming part of a column generation algorithm. For these problems, we need to develop strategies and mechanisms to enable us solve as few subproblems as possible while ensuring good properties of the search. An implementation of the scheme based on a particular scalarization method can also be adapted for another scalarization method.

2.3 Column Search Strategies

Different versions of the generalized column generation scheme in Figure 2 can be defined based on how many subproblems are solved and the strategies used to search for relevant columns. We present three different implementations of the generalized scheme for the case of a BOCOMMO. These implementations can, however, be adapted for other BOCO problems. In what follows, we will denote the LRMP containing only columns $k \in \overline{\Omega}_1$ such that $\sigma_k \leq \varepsilon$ by LRMP(ε).

Point-by-Point Search (PPS).

A standard and intuitive way of implementing the generalized scheme in to solve LRMP(ε) completely for any given value of the parameter ε before changing this value following the process explained in Algorithm 1. That is, for any given value of ε, LRMP(ε) is solved by column generation until the subproblem proposes no new columns that can improve the objective value of LRMP. We call this approach the Point-by-Point Search (PPS) and it is summarized in Algorithm 2. Although PPS is simple and easy to implement, it takes no advantage of the similar subproblems that need to be solved for the different values of ε. The column generation method may also be slow to converge for a given value of ε but PPS requires us to wait for it to converge before moving on to a different value. This can result in a huge number of "irrelevant" columns being added to the RLPM and a long computational time. Moreover, since PPS treats each point of a lower bound set separately, completely different columns are usually used to define each of the points. This is undesirable in certain applications like the one presented by [22] where we want the columns corresponding to different points of a lower bound set to closely resemble each other. The following strategies avoid some of these problems by generating, at each step, columns that are relevant for several values of ε.

Improved Point-by-Point Search (IPPS).

Using heuristics to generate columns can improve the performance of column generation [6]. These heuristics are used to cheaply generate other relevant columns from those found by a subproblem algorithm. Here, we are interested in heuristics that can take advantage of the similar subproblems corresponding

Algorithm 2 Point-by-Point Search (PPS)

Output: A lower bound set **L**.
1: Set $\varepsilon \leftarrow \infty$, and $\mathbf{L} \leftarrow \emptyset$.
2: **while** LRMP(ε) is feasible **do**
3: Solve LRMP(ε) once to obtain a vector of dual values.
4: Let c^* be the optimum, θ^* be the optimal vector, and compute $\sigma^* = \max_{k \in \overline{\Omega}} \{\sigma_k \theta_k^*\}$.
5: Solve the subproblem $S(\varepsilon)$ and let Λ be the set of columns obtained.
6: **if** $\Lambda \neq \emptyset$ **then**
7: Add one or more columns in Λ to LRMP.
8: **else**
9: $\mathbf{L} \leftarrow \mathbf{L} \cup \{(c^*, \sigma^*)\}$ and $\varepsilon \leftarrow \sigma^* - 1$.
10: Set $\theta_k = 0$ for all k such that $\sigma_k > \varepsilon$.
11: **end if**
12: **end while**

to different values of ε. Once a column has been found by using the subproblem algorithm, we wish to quickly generate other columns that are relevant for the current subproblem and may also be relevant for other subproblems. A column which is relevant for a current subproblem may not be relevant for another subproblem since the associated vectors of dual values do not necessarily have the same values. Nevertheless, we can expect that two subproblems that are close in terms of objectives may also be close in terms of the solution of LRMP and therefore close in terms of dual variable values. For this reason, a column generated by an algorithm or a heuristic may also be relevant for several other subproblems apart from the current one. Moreover, standard algorithms used to solve a subproblem are most times only interested in finding the "best" columns for the current subproblem. For this reason, many columns are left out because they are not considered among the "best" for the considered subproblem. This may be desirable for single objective problems but in the bi-objective case, a column which is not so good for one subproblem may be very good for another subproblem. The main idea of IPPS is to improve on the performance of PPS by using heuristics before Step 7 in Algorithm 2 to generate more columns. For a BOCOMMO, we are interested in an algorithm or a heuristics that can take advantage of the redefinition of a column and efficiently search for more columns by modifying the ones found by a subproblem algorithm. IPPS can be useful as a column generation based heuristic since at each iteration it tries to generate a set of columns that are relevant for several subproblems. These columns can be directly inserted in the restricted master problem as they have an impact on at least one subproblem. The algorithm used in searching for more columns obviously depends on the problem being treated. The relevance of a column found by the method is evaluated with respect to the same vector of dual values for which the original column was found. This is a distinctive feature of IPPS in contrast to the other strategy which we describe next.

Solve-Once-Generate-for-All (SOGA).

Another implementation of the generalized scheme that takes advantage of the similar subproblems is summarized in Algorithm 3. We call this approach Solve-Once-Generate-for-All (SOGA) and it starts by generating a set of points based on the current columns in the LRMP without generating any additional columns. After generating a set of points, SOGA continues by solving the subproblem corresponding to a single point. If no relevant columns are returned by the subproblem, the convergence of the point is confirmed so it is saved in the lower bound set. Otherwise, from Step 12 to Step 17, each column found is modified several times by using dual information from the other generated points in order to generate more columns. Unlike in the case of IPPS, the relevance of a column after modification is evaluated with respect to another vector of dual values rather than with respect to the one for which the original unmodified column was found. This guarantees that at each iteration, a set of columns that are relevant for a very large number of points are returned to the RMP. For this reason, SOGA can be very useful in designing column generation based heuristics and metaheuristics. Another advantage that SOGA has over PPS and IPPS is that it solves only one subproblem but generates a set of columns that is guaranteed to be relevant for several subproblems. The main challenge of SOGA comes from the difficulty in combining information from an original column and a vector of dual values in heuristics to search for new columns. This may not always be easy. Just like IPPS, the heuristics used in SOGA depend on the specific problem being treated.

3 Application to the Bi-Objective Multi-Vehicle Covering Tour Problem

We present an application problem to demonstrate the different ideas and approaches discussed in the preceding section. The problem considered is an extension of the covering tour problem [11] namely the bi-objective multi-vehicle covering tour problem (BOMCTP). The covering tour problem (CTP) consists in designing a single route over a subset of locations with the aim of minimizing the length of the route. In addition, each location not visited by the route should lie within a fixed radius from a visited location. The fixed radius is called the *cover distance*. The CTP has a generic application in the design of bi-level transportation networks [3]. This kind of problems seeks to construct a primary route of minimum length on a subset of locations in such a way that all other locations that are not on the primary route can easily reach it. An example of the CTP arises in the problem of choosing where to locate post boxes among a set of candidate sites [18]. The aim of this problem is to minimize the cost of a collection route through all post boxes and also ensure that every user is located within a reasonable distance from a post box. Several other examples of the CTP arise in the domain of humanitarian logistics. For example, in the planning of routes for visiting health care teams in developing

Algorithm 3 Solve-Once-Generate-for-All (SOGA)

Output: A lower bound set $\mathbf{L}$.
1: Set $\varepsilon \leftarrow \infty$ and $\mathbf{L} \leftarrow \emptyset$.
2: **repeat**
3: **while** LRMP(ε) is feasible **do**
4: Solve LRMP(ε) once to obtain a vector of dual values.
5: Let c^* be the optimum and θ^* be the optimal vector.
6: Compute $\sigma^* = \max_{k \in \overline{\Omega}} \{\sigma_k \theta_k^*\}$ and set $\varepsilon \leftarrow \sigma^* - 1$.
7: Set $\theta_k = 0$ for all k such that $\sigma_k > \varepsilon$.
8: **end while**
9: Let $P = \{(c^i, \sigma^i)\}$, $\Pi = \{\pi^i\}$, and $\mathcal{E} = \{\varepsilon^i\}$ be the set of generated points, the corresponding set of dual vectors, and the set of ε values, respectively, obtained from Steps 3 to 8 indexed on $i \in I$ with I the set of indexes of the generated points,
10: Solve the subproblem S(ε^1) corresponding to π^1 and let Λ^1 be the set of columns found.
11: **if** $\Lambda^1 \neq \emptyset$ **then**
12: **for** each column in Λ **do**
13: **for** each vector of dual values $\pi^i \in \Pi$ such that $i > 1$ **do**
14: Use heuristics that combine information from the columns in Λ and the vector π^i to generate some relevant columns corresponding to S(ε^i).
15: **end for**
16: Add found columns to the RLPM.
17: **end for**
18: **else**
19: $\mathbf{L} \leftarrow \mathbf{L} \cup \{(c^i, \sigma^i)\}$ and $\varepsilon \leftarrow \sigma^i - 1$.
20: Set $\theta_k = 0$ for all k such that $\sigma_k > \varepsilon$.
21: **end if**
22: **until** $P = \emptyset$.

countries where medical services can only be delivered to a subset of villages, but all users must be able to reach a visiting medical team [3,15].

A bi-objective generalization of the CTP has been proposed by [17]. The cover distance in the bi-objective CTP (BOCTP) is not fixed in advance but rather induced by the constructed route. It is computed by assigning each non-visited location to the closest visited location and calculating the maximum of the assigned distances. The objectives are to minimize the length of the route and the *induced cover distance*. The authors proposed a two-phase cooperative strategy to solve the problem. This strategy combines a multi-objective evolutionary algorithm with a branch-and-cut algorithm initially designed by [11] to solve the CTP. [13] present a multiple vehicle extension of the CTP namely the multi-vehicle covering tour problem (MCTP). In the MCTP, the combined length of a set of routes, all of which must start from a common location, is minimized for a fixed cover distance. In addition, the number of locations visited by a single route and the length of the route cannot exceed predetermined constants p and q, respectively. The authors proposed an integer linear programming formulation as well as three heuristic methods for the problem. Quite recently, two exact methods have been proposed for the MCTP. The first is a branch-and-price algorithm proposed by [16] whereas the second is a branch-and-cut algorithm proposed by [12]. A metaheuristic for the MCTP was also proposed by [12].

The problem discussed in this section, the bi-objective multi-vehicle covering tour problem (BOMCTP) can be seen as a combination of the BOCTP and the MCTP. The BOMCTP is an interesting problem as it has many features (like multiple routes, restrictions on the tours, optional visits, etc.) that are encountered in difficult VRPs. The subproblem which is an elementary shortest path problems with resource constraints (ESPPRC) is also representative of the one encountered in VRPs solved by column generation. As we will later see, the variant of the ESPPRC encountered for this problem is more difficult that many other variants. The BOMCTP is therefore a good benchmark and we can expect that if the method is successful for it it should also be successful to VRPs specially and also other combinatorial optimization problems.

3.1 Problem Description

The BOMCTP is defined on a graph $G = (V \cup W, E)$ where $V \cup W$ is a set of nodes and E is a set of edges. The nodes of V represent locations which may be visited by a route whereas the members of W are to be assigned to visited nodes of V. There is a subset of nodes $T \subseteq V$, which must be visited by at least one route. In real applications, the members of T represent important locations where we require at least one route to pass. In particular, $v_0 \in T$ is the depot where all routes must start and also end. Set E is made up of edges connecting all pairs of nodes in $V \cup W$ and a distance matrix $D = (d_{ij})$ satisfying the triangle inequality is defined on E. The BOMCTP consists in designing a set of routes over a subset of V which should include all nodes of T. Each route should visit not more than p nodes of $V\backslash\{v_0\}$ and its length must not exceed q where p and q are predetermined constants. The two objectives are to minimize the total length of the set of routes and the cover distance induced by the set.

The cover distance induced by a set of routes (denoted by $\Gamma_{\max}$) is defined as the maximum distance from a node of W to the closest visited node of $V\backslash\{v_0\}$. By definition, the value of d_{ij} for every couple $(v_i, w_j) \in V\backslash\{v_0\} \times W$ is a candidate value for $\Gamma_{\max}$. Nevertheless, some of these candidate values do not correspond to feasible $\Gamma_{\max}$ values. For any given instance of the BOMCTP, we can use an idea similar to the one introduced by [17] for the BOCTP to determine the feasible values of $\Gamma_{\max}$. Since no proof was given by [17], we restate the idea in the following proposition and also give a simple proof.

Proposition 1 *Given $v_i \in V\backslash\{v_0\}$ and $w_j \in W$, d_{ij} is a feasible value for $\Gamma_{\max}$ if and only if the following two conditions are satisfied.*

1. *$\forall v_t \in T\backslash\{v_0\}$ such that $v_t \neq v_i$, $d_{ij} \leq d_{tj}$ and*
2. *$\forall w_l \in W$ such that $w_l \neq w_j$, $\exists v_h \in V\backslash\{v_0\}$ such that $d_{hl} \leq d_{ij} \leq d_{hj}$.*

Proof The proof of this proposition is given in two parts. Part I shows that if d_{ij} is a feasible value for $\Gamma_{\max}$ then conditions (1) and (2) are satisfied. Part II also shows that if conditions (1) and (2) are satisfied then d_{ij} is a feasible value for $\Gamma_{\max}$.

Part I. The necessary conditions for d_{ij} to be the value of $\Gamma_{\max}$ are that v_i is visited by a selected route and w_j is assigned to v_i. Yet, each node of W is assigned to a visited node of $V\backslash\{v_0\}$ that is closest to it. Since all nodes $v_t \in T\backslash\{v_0\}$ are visited in any feasible solution, w_j will be assigned to v_i only if $d_{ij} \leq d_{tj}$. This proves condition (1). Next, suppose that $\Gamma_{\max} = d_{ij}$ but condition (2) is false. That is, for a given $w_l \in W$ such that $w_l \neq w_j$, either $d_{ij} > d_{hj}$ or $d_{ij} < d_{hl}$ for all $v_h \in V\backslash\{v_0\}$. Then, either w_j will be assigned to v_h (since v_h is closer to it than v_i) or if w_j is assigned to v_i then d_{hl} is a better candidate for $\Gamma_{\max}$ (since it is greater than d_{ij} and $\Gamma_{\max}$ is determined by the maximum of the assigned distances). In both cases, d_{ij} cannot be the value for $\Gamma_{\max}$ and so condition (2) must be true if $\Gamma_{\max} = d_{ij}$.

Part II. If conditions (1) and (2) are satisfied then w_j will be assigned to v_i (if it is visited) rather than to a node $v_t \in T\backslash\{v_0\}$ as it is the closest node. If v_i is visited and w_j is assigned to v_i, then condition (2) implies that d_{ij} can possibly be the maximum value among all assigned distances and so a feasible value for $\Gamma_{\max}$.

Clearly, there is a finite number of values for $\Gamma_{\max}$ and by using the two conditions in Proposition 1, all the feasible values of $\Gamma_{\max}$ can be computed for any given instance of the BOMCTP.

3.2 Mathematical Modeling

Let Ω represent the set of all feasible columns. A feasible column $k \in \Omega$ is defined by a route R_k and a subset $\Psi_k \subseteq W$. The route R_k is a Hamiltonian cycle on a subset of V, including the depot, visiting not more than p nodes and of length not exceeding q. The length of R_k is denoted c_k. For each route, we choose a subset $\Psi_k \subseteq W$ of nodes it may cover and define σ_k as the maximum distance between a node of Ψ_k and the closest node of $R_k\backslash\{v_0\}$. We define $a_{ik} = 1$ if column $k \in \Omega$ visits $v_i \in V$ and $a_{ik} = 0$ if this is not the case. The constant $b_{jk} = 1$ if $w_j \in \Psi_k$ and $b_{jk} = 0$ otherwise. The binary variable θ_k is used to indicate whether column $k \in \Omega$ is selected in a solution ($\theta_k = 1$) or not ($\theta_k = 0$). The BOMCTP can be described as:

$$\text{Minimize} \quad \sum_{k\in\Omega} c_k\theta_k \tag{12}$$

$$\text{Minimize} \quad \Gamma_{\max} \tag{13}$$

subject to:

$$\sum_{k\in\Omega} a_{ik}\theta_k \geq 1 \qquad (v_i \in T\backslash\{v_0\}), \tag{14}$$

$$\sum_{k\in\Omega} b_{jk}\theta_k \geq 1 \qquad (w_j \in W), \tag{15}$$

$$\Gamma_{\max} - \sigma_k\theta_k \geq 0 \qquad (k \in \Omega), \tag{16}$$

$$\theta_k \in \{0,1\} \qquad (k \in \Omega). \tag{17}$$

The objectives of minimizing the length of the set of routes and the induced cover distance are given in (12) and (13), respectively. Constraints (14) ensure that each node of $T\backslash\{v_0\}$ is visited by at least one selected route. The fact that each node of W should be assigned to a visited node of $V\backslash\{v_0\}$ is indicated by Constraints (15). Finally, Constraints (16) ensure that the value of the induced cover distance conforms to its definition.

By following the idea presented in Section 2 to reformulate a BOCOMMO, we define a new set of feasible columns $\overline{\Omega}$ where the feasibility of a column k in $\overline{\Omega}$ depends not just on R_k but also on σ_k. The resulting master problem, MP, is given by:

$$\text{Minimize} \quad \sum_{k\in\overline{\Omega}} c_k\theta_k \tag{18}$$

$$\text{subject to:} \quad \sum_{k\in\overline{\Omega}} a_{ik}\theta_k \;\geq\; 1 \qquad (v_i \in T\backslash\{v_0\})\,, \tag{19}$$

$$\sum_{k\in\overline{\Omega}} b_{jk}\theta_k \;\geq\; 1 \qquad (w_j \in W)\,, \tag{20}$$

$$\theta_k \;\in\; \{0,1\} \qquad (k \in \overline{\Omega})\,. \tag{21}$$

We recall that the second objective to minimize $\Gamma_{\max}$ and the Constraints (16) do not appear in the above formulation due to the redefinition of a column. The restricted master problem, RMP, is obtained by restricting MP to a subset $\overline{\Omega}_1$ of $\overline{\Omega}$ and rewriting Constraints (21) as $\theta_k \geq 0$ for $k \in \overline{\Omega}$. The dual problem of MP is given by:

$$\text{Maximize} \quad \sum_{v_i\in T\backslash\{v_0\}} \pi_i + \sum_{w_j\in W} \beta_j$$

$$\text{subject to:} \quad \sum_{v_i\in T\backslash\{v_0\}} a_{ik}\pi_i + \sum_{w_j\in W} b_{ik}\beta_j \;\leq\; c_k \quad (k \in \Omega)\,,$$

$$\pi_i \;\geq\; 0 \quad (v_i \in T\backslash\{v_0\})\,,$$

$$\beta_j \;\geq\; 0 \quad (w_j \in W)\,,$$

where π_i for $v_i \in T\backslash\{v_0\}$ and β_j for $w_j \in W$ are the vectors of non-negative dual values associated with Constraints (19) and (20), respectively. From this dual formulation, we can deduce the subproblem $S(\varepsilon)$ corresponding to LRMP(ε) as finding columns k in $\overline{\Omega}\backslash\overline{\Omega}_1$ that satisfy the condition

$$c_k - \sum_{v_i\in T\backslash\{v_0\}} \pi_i a_{ik} - \sum_{w_i\in W} \beta_j b_{ik} \;<\; 0\,.$$

We define $\pi_i^* = \pi_i$ if $v_i \in T\backslash\{v_0\}$ and $\pi_i^* = 0$ if $v_i \in V\backslash T$. We also let $x_{ijk} = 1$ if route R_k visits v_j immediately after visiting v_i and $x_{ijk} = 0$ otherwise. Note

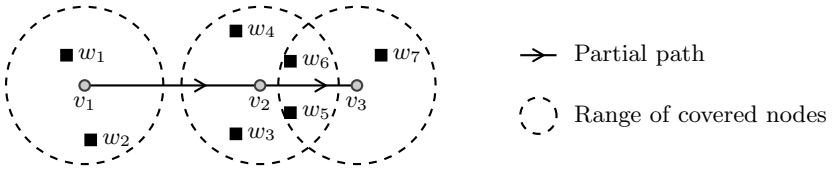


Fig. 3: Non-Additive Nature of the Subproblem.
The reduced cost of the partial path $v_1 \rightarrow v_2 \rightarrow v_3$ is different from the sum of the reduced costs of the partial paths $v_1 \rightarrow v_2$ and $v_2 \rightarrow v_3$.

that $c_k = \sum_{(v_i,v_j)\in V^2} d_{ij}x_{ijk}$ and $a_{ik} = \sum_{(v_i,v_j)} x_{ijk}$ so the subproblem $S(\varepsilon)$ can be written as:

$$\text{Minimize} \sum_{(v_i,v_j)\in V^2} (d_{ij} - \pi_i^*)x_{ijk} - \sum_{w_i\in W} \beta_j b_{ik} \text{ subject to } k \in \overline{\Omega}\backslash\overline{\Omega}_1\,. \quad (22)$$

Given that $R_k \subseteq V$ whereas $\Psi_k \subseteq W$, we need to construct a route on a subset of V with the aim of minimizing its cost $\sum_{(v_i,v_j)\in V^2} (d_{ij} - \pi_i^*)x_{ijk}$ and also choose a subset of nodes $w_j \in W$ with the aim of maximizing the profits, β_j, associated to its members. The profit associated to a node of W can be collected at most once on any single route even though different nodes of the route R_k may be able to cover it. Problem (22) is therefore a non-additive ESPPRC. That is, the cost of a partial path is not necessarily the same as the sum of the costs of its subpaths. An example is given in Figure 3 where we suppose that $d_{12} = 2$, $d_{23} = 1$, and $\beta_j = 1$ for $j = 1, 2, \ldots, 7$. The reduced costs of the partial paths $v_1 \rightarrow v_2 \rightarrow v_3$, $v_1 \rightarrow v_2$, and $v_2 \rightarrow v_3$ are given by $3 - 7 = -4$, $2 - 6 = -4$, and $1 - 5 = -4$, respectively. That is, the reduced cost of $v_1 \rightarrow v_2 \rightarrow v_3$ is not the same as the sum of the reduced costs of its constituent partial paths because nodes w_5 and w_6 can be covered by both v_2 and v_3. [20] discuss non-additive shortest path problems and also present some real life applications.

3.3 Solving the Subproblem $S(\varepsilon)$

A dynamic programming algorithm [9] accelerated by means of the decremental state space relaxation (DSSR) algorithm [21,2] is used in solving the subproblem. The DSSR principle is that first a relaxed problem is solved allowing to generate a path containing cycles. The nodes that are visited several times are then considered critical and the problem is solved ensuring that these additional constraints are enforced. This is iterated until the solution of the relaxed problem is cycle free as it is also an optimal solution for the problem with the elementary constraints.

Two resources are considered when implementing the algorithm for this problem. The first ressource is the number of nodes a route may visit and the second one is the length of the route. There resources are limited to a maximum of p and q, respectively. A label $\Lambda^i = (\tilde{c}^i, \tilde{p}^i, \tilde{q}^i)$ is used to represent a partial

path from the depot v_0 to node v_i. The components of Λ^i are the reduced cost up to the current node, $\tilde{c}^i$, the total number of nodes of $V\backslash\{v_0\}$ visited, $\tilde{p}^i$, and the length of the partial path, $\tilde{q}^i$. In order to simplify the notation, some components of a label will not be described here in details. The label store information about the fact that some nodes of W can be covered by an unvisited node in $V \setminus \{v_0\}$, i.e. the distance between the two nodes is less than ε, and that it is profitable to do so. This is done by keeping a boolean set on the nodes of W that still need to be covered, which is updated when a node of V is visited during the label extension. Another aspect of the label not described in details here is how we check that a path is elementary or not, and if a node is a critical node in the DSSR scheme. In the implementation, it is done using a set. These implementation are straightforward and the validity of our discussion is not affected by this omission.

Label Extension.

When extending a label from a node $v_i \in V$ to another node $v_j \in V$, nodes of W not yet covered by the label but which can be covered by v_j are identified and the resulting profit is subtracted from the current reduced cost of the label. Doing so ensures that we obtain the minimum possible reduced cost for each label without counting the profit associated to any node of W more than once. That is, even when the elementary condition is relaxed in the DSSR algorithm, there is no interest in visiting a node of $V\backslash T$ more than once since the reduced cost of the path resulting from the second visit will be worse than the one from the first visit. The other ressources are straightforward to compute.

Dominance Rule.

Given any two labels $\Lambda_1^i = (\tilde{c}_1^i, \tilde{p}_1^i, \tilde{q}_1^i)$ and $\Lambda_2^i = (\tilde{c}_2^i, \tilde{p}_2^i, \tilde{q}_2^i)$ on node $v_i \in V$, we say that Λ_1^i dominates Λ_2^i if and only if $\tilde{c}_1^i \leq \tilde{c}_2^i - F_{12}$, $\tilde{p}_1^i \leq \tilde{p}_2^i$, $\tilde{q}_1^i \leq \tilde{q}_2^i$, and at least one of the inequalities is strict. The factor F_{12} represents the sum of the profits associated to nodes of W, i.e. the dual variable values, that are covered by Λ_1^i but not yet covered by Λ_2^i. This factor is necessary to account for the non-additive nature of the ESPPRC as demonstrated by Figure 4. In this figure, the label Λ_1^1 represents a partial path from v_0 to v_1 that has already visited v_2 whereas Λ_2^1 is another partial path that has not yet visited v_2. That is, Λ_1^1 can no longer visit v_2 but it is possible to extend Λ_2^1 to v_2. The total profit that can be collected (from covering some nodes of W) by visiting node v_2 is 4. For simplicity, we suppose that no node of W can be covered by more than one node of V. The figure shows all the labels that are generated when no dominance rule is applied. Three labels arrive at the copy of the depot, v_d. Out of these three labels, only Λ_1^d and Λ_3^d are actually of interest since they are not dominated by any other labels. Without the factor F_{12}, we would discard Λ_2^1 because it is dominated by Λ_1^1 and so the nondominated label Λ_3^d would not be generated. Thus, the factor F_{12} ensures that no label that can lead to a nondominated path is eliminated.

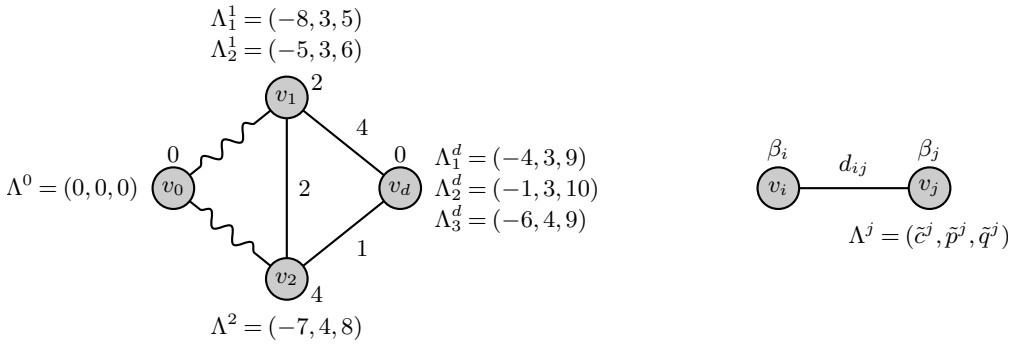


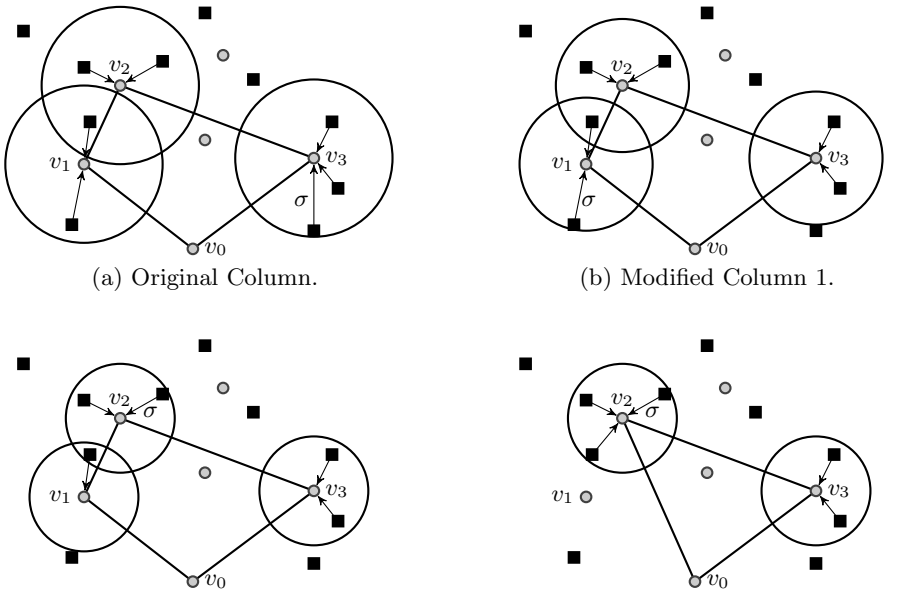
Fig. 4: Dominance Relationship between Labels.

Node v_d is a copy of the depot, v_0. Λ_1^1 represents the partial path $v_0 \to v_3 \to v_2 \to v_1$ and Λ_2^1 represents $v_0 \to v_3 \to v_4 \to v_1$. Λ_1^d and Λ_2^d are direct extensions of Λ_1^1 and Λ_2^1, respectively, to v_d. Λ^2 is a direct extension of Λ_2^1 to v_2.

3.4 Implementation of Column Search Strategies

Among the three strategies presented in the preceding section, PPS is the simplest to implement. For a given value of ε, we solve LRMP(ε) by column generation until it converges. We need to ensure that only columns k for which $\sigma_k \leq \varepsilon$ are allowed in the solution of LRMP(ε). We also need to ensure that a visited node of $V \backslash \{v_0\}$ can only cover a node of W that lies in a radius of ε from it. The other two strategies IPPS and SOGA incorporate heuristics that are dependent on the specific problem being treated and we present examples of these heuristics for the BOMCTP. In the description of IPPS and SOGA, we will let $k' := (R'_k, \Psi'_k)$ be a column returned by the DSSR algorithm after solving the subproblem $S(\varepsilon')$. The vectors of dual values used by the DSSR algorithm in obtaining column k' will also be denoted by π' and β'.

IPPS Heuristic for the BOMCTP. When solving $S(\varepsilon')$, the DSSR algorithm constructs a column $k' := (R'_k, \Psi'_k)$ by taking Ψ'_k to be all the nodes of W that lie within a radius of ε' from a node of $R'_k \backslash \{v_0\}$. This is to ensure that the reduced cost of k' is minimized. Nevertheless, Ψ'_k does not necessarily need to include all the nodes of W that can be covered by R'_k. Indeed, Ψ'_k can be chosen to be any subset of W each of which lie within a radius of ε' from a node of $R'_k \backslash \{v_0\}$ and such that the sum of the profits β' associated with this subset exceeds the cost c_k. A column defined in this way is hardly returned by the DSSR algorithm for the current subproblem since it will be dominated by another column defined by the same route, but covers some more nodes of W. The IPPS heuristic for the BOMCTP relies on this observation. The details of how the heuristic works is depicted by Figure 5. A column k' is successively modified by removing the node of Ψ'_k that induces the value of σ'_k (that is, which is farthest from the closest node of R_k) in order to create another column $k'' := (R''_k, \Psi''_k)$ where $\sigma''_k < \sigma'_k$. The reduced cost of k'' is evaluated with respect to the same vectors of dual values π' and β' which were used by



(a) Original Column. (b) Modified Column 1.

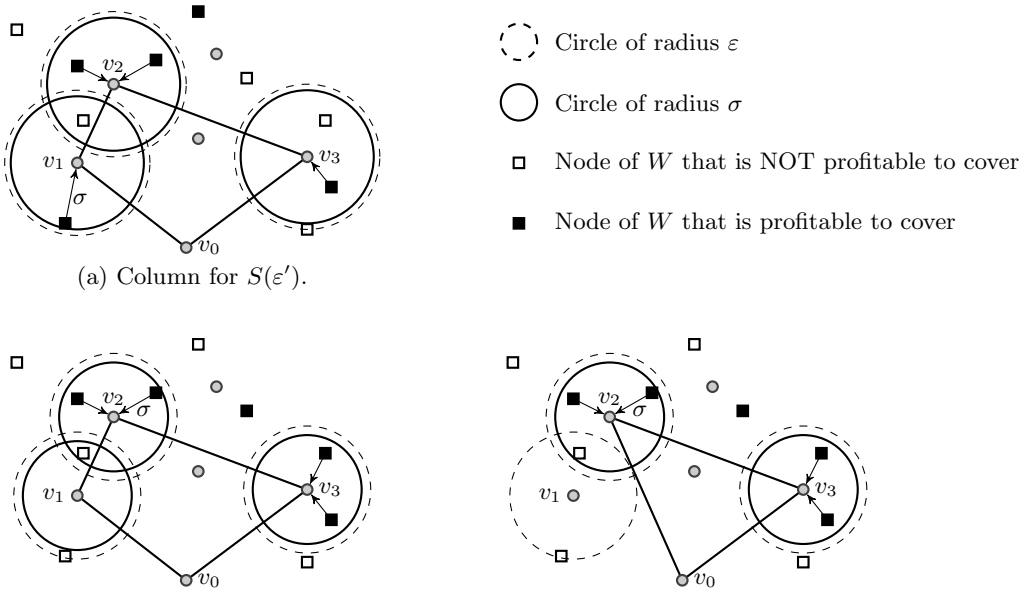
(c) Modified Column 2. (d) Modified Column 2 (improved).

Fig. 5: IPPS Heuristic for the BOMCTP.
A column is successively modified by removing the node of W that induces the value of σ in order to generate several other columns.

the DSSR algorithm in solving $S(\varepsilon')$. This means that if the reduced cost of k'' is negative, then it is guaranteed to be relevant for $S(\varepsilon')$. The relevance of k'' for another value of $\varepsilon \neq \varepsilon'$ is not guaranteed by this heuristic. Initially, $c''_k = c'_k$ as in Figure 5 but if a node $v_i \in R''_k \backslash \{v_0\}$ does not uniquely cover at least one node of Ψ''_k, then it is removed from R''_k in order to have $c''_k < c'_k$ and further minimize the reduced cost as shown in Figure 5 d. The successive modifications end when no more columns having negative reduced costs can be obtained. The whole procedure is summarized in Algorithm 4.

Algorithm 4 IPPS Heuristic for the BOMCTP

Input: An original column $k' := (R'_k, \Psi'_k)$.
while k' has negative reduced cost and $\Psi'_k \neq \emptyset$ **do**
  Create a column $k'' := (R''_k, \Psi''_k)$ where $R''_k = R'_k$ and $\Psi''_k = \Psi'_k$.
  Remove the node $w_j \in \Psi''_k$ that induces the value of σ''.
  Recalculate the value of σ''_k.
  Delete any node $v_i \in R''_k \backslash \{v_0\}$ that does not uniquely cover (based on the value of σ''_k) at least one profitable node of Ψ''_k.
  Recalculate the reduced cost of k''.
  if k'' has negative reduced cost **then**
    Add k'' to the RLPM.
  end if
  $k' \leftarrow k''$.
end while



(a) Column for $S(\varepsilon')$.

(b) Column for $S(\varepsilon'')$ where $\varepsilon'' < \varepsilon'$.

(c) Improved Column from (b).

Fig. 6: SOGA Heuristic for the BOMCTP.
A new column is constructed from an original column by incorporating dual values corresponding to another subproblem.

SOGA Heuristic for the BOMCTP. Suppose that π'' and β'' are the vectors of dual values corresponding to RMP(ε'') where $\varepsilon'' \neq \varepsilon'$. Note that in general $\pi'' \neq \pi'$ and $\beta'' \neq \beta'$. The principle of a SOGA heuristic for the BOMCTP which is demonstrated in Figure 6 is to modify $k' := (R'_k, \Psi'_k)$ to obtain another column $k'' := (R''_k, \Psi''_k)$ by completely reconstructing the set of nodes that may be covered (Ψ''_k). The set Ψ''_k is constructed by taking all profitable nodes within a radius of ε'' from a node of $R''_k \backslash \{v_0\}$. The profit associated with covering a node of W depends on β'' rather than on β'. In other words, the reduced cost of the modified column k'' is evaluated with respect to the dual vectors π'' and β''. This means that if the reduced cost of k'' is negative, then it is guaranteed to be relevant for $S(\varepsilon'')$ but probably not for $S(\varepsilon')$. After constructing Ψ''_k, we compute $\sigma''_k = \max\{d_{ij} : v_i \in R''_k \text{ and } w_j \in \Psi''_k\}$. A node $v_i \in R''_k \backslash \{v_0\}$ that does not uniquely cover a profitable node of W is removed from R''_k in order to reduce the length of the route and further minimize the reduced cost (see Figure 6 c). Finally, all the other non-profitable nodes of W that lie within a radius of σ''_k from a node of $R''_k \backslash \{v_0\}$ are added to Ψ''_k. Algorithm 5 summarizes this heuristic.

Algorithm 5 SOGA Heuristic for the BOMCTP

Input: An original column $k' := (R'_k, \Psi'_k)$ and a set of dual variable vectors together with the corresponding values of ε for which they were computed.

if k' has negative reduced cost and $\Psi'_k \neq \emptyset$ **then**
  for each vector of dual variables **do**
    Create a column $k'' := (R''_k, \Psi''_k)$ where $R''_k = R'_k$ and Ψ''_k contains all profitable nodes of W that can be covered (based on the corresponding value of ε) by a node of $R''_k \backslash \{v_0\}$.
    Recalculate the value of σ''_k.
    Delete any node $v_i \in R''_k \backslash \{v_0\}$ that does not uniquely cover (based on the value of σ''_k) at least one node of Ψ''_k.
    Recalculate the reduced cost of k''.
    if k'' has negative reduced cost **then**
      Add all non-profitable nodes of W that lie in a radius of σ''_k from a node of $v_i \in R''_k \backslash \{v_0\}$ to Ψ''_k.
      Add k'' to the RLPM.
    end if
  end for
end if

4 Computational Results

We present results from experiments conducted to evaluate the quality of lower bound sets computed for the BOMCTP when we use a standard ε-constraint formulation and when we use each of the three strategies based on the reformulation of a BOCOMMO. In order to better evaluate the lower bound sets, we compute corresponding upper bound sets. Upper bound sets can be computed using any known methods (exact, heuristics, metaheuristics) to search for a set of feasible points that do not dominate one another. In this work, we use a very simple heuristic to compute an upper bound set after computing a lower bound set by column generation. The idea is to solve RMP (the integer program) several times by following the idea of Algorithm 1. That is, we consider the RMP with the columns it contains after computing a lower bound set and follow Algorithm 1 by replacing the words "scalarized problem" with RMP. Although an upper bound set is made up of feasible points, they do not necessary belong to the nondominated set $\mathcal{Y}_N$ since the RMP may not contain all relevant columns needed to define $\mathcal{Y}_N$. Nevertheless, if the columns in RMP are "good" for MP then we expect that the obtained upper bound set will be a good approximation of $\mathcal{Y}_N$.

4.1 Evaluation of Bound Sets

We use two measures presented by [7] in evaluating the quality of the computed bound sets. The first measure (μ_1) is distance based whereas the second measure (μ_2) is area based. Roughly speaking, μ_1 represents the worst distance (with respect to the range of objective values) between a point of the upper bound set and a point of the lower bound set closest to it. Also, μ_2 represents the fraction of the area that is dominated by the lower bound set but not by the

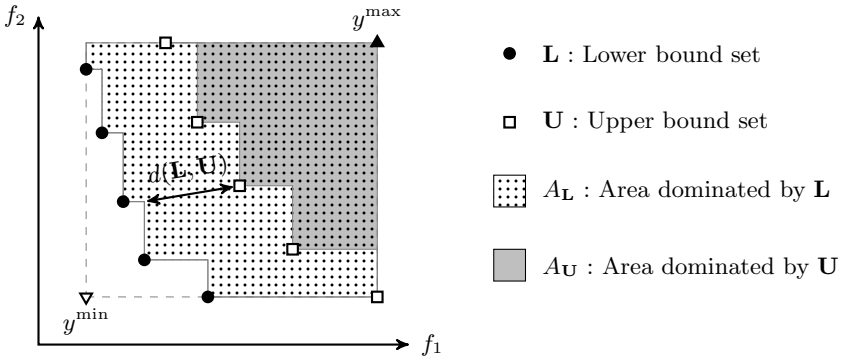


Fig. 7: Calculation of Quality Measures.

upper bound set. This is, the area where additional points of $\mathcal{Y}_N$ can be found. Given a lower bound set $\mathbf{L}$ and a corresponding upper bound set $\mathbf{U}$, we let $y_i^{\max}$ for $i \in \{1, 2\}$ be the maximum value of the i^{th} objective when we consider the union of $\mathbf{L}$ and $\mathbf{U}$. In the same way, we let $y_i^{\min}$ for $i \in \{1, 2\}$ be the minimum value of the i^{th} objective when we consider the union of $\mathbf{L}$ and $\mathbf{U}$. We define the points $y^{\max} := (y_1^{\max}, y_2^{\max})$ and $y^{\min} := (y_1^{\min}, y_2^{\min})$. The distance between $\mathbf{L}$ and $\mathbf{U}$ (denoted by $d(\mathbf{L}, \mathbf{U})$) is defined as the maximum of the minimum distances by which we need to displace a point of $\mathbf{U}$ so that it is not dominated by any point of $\mathbf{L}$. Next, let $A_{\mathbf{L}}$ and $A_{\mathbf{U}}$ be the areas of the regions in the rectangle with $y^{\min}$ and $y^{\max}$ at opposite corners that are dominated by $\mathbf{L}$ and $\mathbf{U}$, respectively. The two measures are given by

$$\mu_1 := \frac{d(\mathbf{L}, \mathbf{U})}{\|y^{\max} - y^{\min}\|_2} \quad \text{and} \quad \mu_2 := \frac{A_{\mathbf{L}} - A_{\mathbf{U}}}{A_{\mathbf{L}}},$$

where $\|\cdot\|_2$ is the Euclidean norm. Figure 7 gives a visual representation of how the measures are calculated. These two measures complement each other and as explained by [7], they can be seen to play a role similar to the optimality gap in single objective optimization. If a lower bound set and a corresponding upper bound set are good, then we expect that both μ_1 and μ_2 will be small in value. The smaller both values are, the better the quality of the bound sets.

4.2 Description of Instances and Experiments

Random instances similar to those described in the literature [11,13,17] were used for the experiments. The node sets were obtained by generating $|V| + |W|$ points in the $[0, 100] \times [0, 100]$ square with the depot restricted to lie in $[25, 75] \times [25, 75]$. Set T (respectively, V) is taken to be the first $|T|$ (respectively, $|V|$) points and set W is taken as the remaining points. The distance between two points is calculated as the Euclidean distance between them. Five instances for every combination of $|V| \in \{30, 40, 50\}$ and $|W| \in \{2|V|, 3|V|\}$ were generated.

Values of $|T|$ in $\{1, \lceil 0.25 \cdot |V| \rceil, \lceil 0.50 \cdot |V| \rceil\}$, p in $\{5, 8\}$, and $q = \infty$ were tested. All computer codes were written in C/C++ and the linear programs were solved with ILOG CPLEX 12.4. The tests were run on an Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93GHz computer with a 2 GiB RAM. A Summary of the results obtained are given in Tables 1 and 2. All the values in the tables are averages over five instances. In these tables, "standard" refers to a BOMCTP formulation based on a standard ε-constraint method (by directly adding constraints of the form $\Gamma_{\max} \leq \varepsilon$ to the formulation). Such a formulation can be found in [24]. Table 1 gives the quality of the computed bound sets whereas Table 2 gives the computational times and other characteristics of the column generation method. Given that column generation is an exact method for an LMP based on a particular formulation, the same lower bound set is obtained for a given instance based on the formulation no matter the column search approach used. For each instance, the number of elements in the lower bound sets obtained from the "standard" formulation and the reformulation are given under the columns with headings $|\mathbf{L}|$ and $|\mathbf{L}^*|$, respectively. The number of elements in an upper bound set is given under the columns with headings $|\mathbf{U}|$. The upper bound sets obtained for each instance are different for the different formulations as well as the different column search strategies. This is because each strategy generates different columns when computing a lower bound set. The columns with headings *time*, *dssr*, and *cols* represent the computational times (in cpu seconds), the number of times the sub-problem was solved with the DSSR algorithm and the total number of columns generated, respectively. Values of the two quality indicators are expressed as percentages under the columns $\mu_1\%$ and $\mu_2\%$.

4.3 Summary of Results

It can be seen from Table 1 that the bound sets obtained from the reformulated model are significantly better than those obtained from a model that uses a standard ε-constraint approach. This is seen by comparing the values of μ_1 and μ_2 for PPS, IPPS, and SOGA with their counterparts from "Standard". For example, a standard ε-constraint approach obtained the values ($\mu_1\% = 4.5, \mu_2\% = 24.8$) for the instance $p = 5$, $|T| = 1$, $|V| = 50$, $|W| = 150$ whereas those for PPS, IPPS, and SOGA were ($\mu_1\% = 0.2, \mu_2\% = 0.5$), ($\mu_1\% = 0.2, \mu_2\% = 0.7$), and ($\mu_1\% = 0.2, \mu_2\% = 0.4$), respectively. There is not much difference between the values of $\mu_1\%$ and $\mu_2\%$ for the different search strategies (PPS, IPPS, and SOGA) based on the reformulated model so no preference for a particular column search approach can be established from these values. For both the standard and reformulated models, there is a general increase in the values of $\mu_1\%$ and $\mu_2\%$ when $|V|$ increases and also when $|T|$ increases. This trend is expected since increasing the values of $|V|$ and $|T|$ usually increases the computational effort needed to solve an instance.

From Table 2, we see that computing bound sets based on the standard model is faster than using the reformulated model. This is not so surprising

Table 1: Quality of Bound Sets for the BOMCTP.

				Standard				PPS				IPPS			SOGA		
p	$\|T\|$	$\|V\|$	$\|W\|$	$\|\mathbf{L}\|$	$\|\mathbf{U}\|$	$\mu_1\%$	$\mu_2\%$	$\|\mathbf{L}^*\|$	$\|\mathbf{U}\|$	$\mu_1\%$	$\mu_2\%$	$\|\mathbf{U}\|$	$\mu_1\%$	$\mu_2\%$	$\|\mathbf{U}\|$	$\mu_1\%$	$\mu_2\%$
5	1	30	60	64	29	5.2	26.2	25	26	0.3	0.6	26	0.4	0.6	26	0.4	0.6
5	1	30	90	66	29	6.0	27.6	21	21	0.8	2.3	22	0.9	2.0	22	0.9	2.0
5	1	40	80	58	30	5.5	26.1	26	26	0.3	0.4	27	0.3	0.5	26	0.3	0.5
5	1	40	120	67	37	5.1	25.4	27	28	0.4	1.2	29	0.5	1.1	29	0.5	1.2
5	1	50	100	65	34	4.9	24.7	32	33	0.3	0.7	33	0.2	0.6	33	0.3	0.6
5	1	50	150	67	35	4.5	24.8	30	30	0.2	0.5	30	0.2	0.7	31	0.2	0.4
5	8	30	60	15	9	7.7	54.1	10	9	0.2	5.9	9	0.2	5.7	9	0.1	9.3
5	8	30	90	17	10	11.0	53.3	10	10	2.3	9.6	10	2.3	6.5	10	2.6	6.5
5	10	40	80	15	11	1.4	58.1	10	11	0.8	9.2	11	1.0	9.9	11	1.3	9.6
5	10	40	120	16	13	9.1	61.7	12	12	0.7	7.8	12	0.8	7.8	13	1.1	8.0
5	13	50	100	18	11	16.7	72.9	11	11	0.7	4.5	10	1.1	4.0	10	0.6	3.9
5	13	50	150	19	10	21.6	72.6	8	8	3.3	11.9	8	1.9	10.1	9	3.6	9.8
5	15	30	60	2	6	20.3	54.7	6	5	2.2	18.0	5	2.2	13.7	5	2.2	14.5
5	15	30	90	5	6	40.5	55.7	5	6	1.9	15.7	6	1.4	14.5	6	1.8	15.0
5	20	40	80	6	4	57.3	61.4	5	4	5.5	41.2	4	5.4	40.3	4	4.9	40.0
5	20	40	120	7	5	31.2	62.3	5	5	0.8	38.2	5	0.8	46.4	5	0.8	41.3
5	25	50	100	6	4	24.3	76.8	4	4	18.7	36.3	4	1.1	47.8	4	1.1	51.3
5	25	50	150	9	4	28.9	79.1	3	3	3.0	57.7	3	1.4	41.1	3	1.8	35.9
8	1	30	60	64	29	5.3	26.3	25	26	0.1	0.6	25	0.1	0.5	25	0.1	0.5
8	1	30	90	63	30	6.5	28.2	22	22	0.7	1.4	23	0.8	1.2	22	0.8	1.1
8	1	40	80	58	28	5.6	26.0	27	27	0.1	0.4	27	0.1	0.4	27	0.1	0.4
8	1	40	120	65	35	5.2	25.7	29	30	0.2	0.7	30	0.3	0.7	30	0.3	0.7
8	1	50	100	65	34	4.9	24.7	32	32	0.2	0.7	33	0.2	0.6	33	0.2	0.7
8	1	50	150	64	36	4.7	24.8	30	30	0.3	0.7	30	0.2	0.5	31	0.3	0.5
8	8	30	60	15	10	10.2	65.1	10	10	2.5	8.9	9	1.6	9.8	9	2.5	10.1
8	8	30	90	15	9	28.7	69.5	9	9	2.7	10.7	9	2.7	10.4	10	2.7	10.3
8	10	40	80	14	8	16.6	68.0	8	7	6.9	19.6	9	7.6	26.8	8	7.3	24.6
8	10	40	120	17	11	9.3	73.4	11	11	0.7	7.1	12	1.3	8.8	12	1.1	8.8
8	13	50	100	16	10	35.2	74.2	10	10	3.8	10.9	10	3.8	11.7	10	3.4	10.7
8	13	50	150	17	9	30.1	76.7	8	8	2.4	12.0	8	2.2	10.3	8	2.1	10.0
8	15	30	60	2	6	29.0	65.4	6	5	0.8	26.6	5	1.0	26.2	5	1.5	27.6
8	15	30	90	2	6	35.4	71.1	5	6	2.8	21.5	6	3.2	23.4	6	3.3	23.5
8	20	40	80	2	4	41.2	72.9	4	4	15.4	75.6	4	32.5	74.8	4	15.4	69.3
8	20	40	120	4	6	9.2	73.8	5	5	3.2	50.2	5	3.4	51.7	5	3.8	50.3
8	25	50	100	3	5	34.9	81.3	4	4	2.1	56.1	4	2.2	56.1	4	2.1	56.4
8	25	50	150	4	5	24.3	87.9	3	3	2.5	64.9	3	2.6	71.5	3	3.0	65.7

given the relatively inferior quality of the bound sets obtained from the standard model when compared to those from the reformulated model (see Table 1). There is a clear preference between the column search strategies for the reformulated model. SOGA is clearly the best, followed by IPPS and then PPS. Moreover this preference becomes more evident for difficult instances. For example, the computational times for PPS, IPPS and SOGA for the instance $p = 8$, $|T| = 13$, $|V| = 50$, $|W| = 150$ are 2226.0, 1938.3, and 1033.4, respectively, whereas those for the instance $p = 8$, $|T| = 25$, $|V| = 50$, $|W| = 150$ are 3143.1, 2941.0, and 1731.5, respectively. In addition, SOGA usually needs to solve fewer number of

Table 2: Computational Times for the BOMCTP.

				Standard			PPS			IPPS			SOGA		
p	$\|T\|$	$\|V\|$	$\|W\|$	time	dssr	cols	time	dssr	cols	time	dssr	cols	time	dssr	cols
5	1	30	60	7.8	160	358	18.0	185	1198	13.9	124	1809	13.0	112	1229
5	1	30	90	12.8	93	332	15.4	163	1063	14.6	120	1569	12.3	106	1041
5	1	40	80	15.0	72	416	49.5	228	1597	40.5	155	2099	36.7	141	1610
5	1	40	120	16.0	129	435	126.8	330	2571	94.0	201	3388	86.4	174	2348
5	1	50	100	26.0	152	570	205.8	390	3035	153.9	226	3459	154.5	213	2871
5	1	50	150	24.7	137	591	392.5	486	4053	287.0	247	4054	268.1	224	3087
5	8	30	60	6.3	165	808	13.6	145	1155	10.9	90	1478	9.1	72	982
5	8	30	90	15.3	143	837	17.9	139	1118	14.3	100	1348	11.9	64	842
5	10	40	80	18.6	157	969	41.3	191	1600	36.0	138	2105	27.2	83	1229
5	10	40	120	28.2	195	1048	104.2	182	2441	82.9	199	2441	58.2	101	1758
5	13	50	100	42.4	243	1160	148.6	309	2728	123.6	231	3299	87.5	110	2207
5	13	50	150	51.7	286	1395	186.8	264	2342	174.3	228	2755	102.8	87	1826
5	15	30	60	16.8	149	825	22.9	171	1463	17.6	113	2750	11.4	55	1444
5	15	30	90	26.4	153	973	29.5	141	1198	27.0	114	2067	17.7	51	1083
5	20	40	80	35.8	168	1232	93.0	207	1832	75.0	156	3726	46.0	67	2143
5	20	40	120	53.5	192	1568	143.3	236	2136	127.1	198	3369	67.7	67	2445
5	25	50	100	55.5	227	1913	231.3	257	2363	216.0	219	4193	112.6	75	2886
5	25	50	150	68.2	285	2194	264.4	185	1700	259.9	171	2902	146.9	65	2440
8	1	30	60	11.8	245	374	49.8	227	1627	29.7	152	2235	25.5	136	1657
8	1	30	90	14.9	144	357	31.3	215	1552	23.3	142	2010	18.7	122	1284
8	1	40	80	54.0	124	436	113.4	302	2283	103.6	218	2961	86.8	183	2253
8	1	40	120	47.2	154	438	511.2	481	3949	503.9	293	4663	326.6	243	3652
8	1	50	100	96.6	189	618	1343.7	522	4306	1012.5	335	5071	821.2	289	4393
8	1	50	150	86.5	199	566	1525.2	672	5799	1186.2	384	6005	1042.1	306	4782
8	8	30	60	52.1	231	738	53.6	231	2027	44.6	158	2466	31.9	123	1952
8	8	30	90	58.0	195	953	77.7	219	1873	65.2	154	2071	50.7	101	1558
8	10	40	80	81.4	206	1087	213.4	298	2730	202.2	214	3560	149.6	119	2236
8	10	40	120	69.5	237	988	797.0	475	4427	623.5	320	4102	407.7	183	3837
8	13	50	100	123.7	273	1439	1142.5	571	5379	973.4	413	6220	755.8	216	4749
8	13	50	150	110.4	294	1712	2226.0	545	5119	1938.3	446	6060	1033.4	161	3939
8	15	30	60	42.1	259	952	182.9	333	3086	128.7	205	5493	87.5	103	3298
8	15	30	90	60.3	198	1071	380.6	288	2691	283.4	209	4675	213.6	102	2451
8	20	40	80	153.7	281	1209	1715.3	430	4129	1459.5	318	8396	978.6	120	5614
8	20	40	120	118.7	296	1841	3202.5	564	5392	2973.4	455	8532	1568.9	149	6162
8	25	50	100	209.9	317	2058	7861.5	627	6022	6963.5	538	11026	3917.6	165	8249
8	25	50	150	264.9	334	2314	3143.1	374	3623	2941.0	342	6757	1731.5	130	5438

subproblems than both PPS and IPPS as it can be seen from the columns with headings *dssr*. This is another very good statistic in favor of SOGA since a greater percentage of the computational time in a column generation algorithm is spent on solving the subproblems.

5 Conclusions

This paper investigates the design of efficient column generation algorithms for bi-objective combinatorial optimization problems. We proposed a generalized column generation scheme for this class of problems and presented three specific implementations for the case where one objective is a min-max objective function (BOCOMMO). Instead of directly adding constraints on an objective when modeling a BOCOMMO, we rather use a variant of the ε-constraint method that redefines the set of feasible columns to take the objective into account. By doing so, we keep the strength of the model at the expense of having a possibly more difficult problem. The advantages of using the reformulated model is clearly seen from the quality of the bounds obtained for the bi-objective multi-vehicle covering tour problem. The results obtained from computational experiments also show that significant speedups can be achieved when computing lower bound sets by column generation if the search for columns is intelligently managed. Given that the time needed to compute such quality bound sets can be very long, future works are aimed at finding a good compromise between the quality of bound sets and the computational time. It is also necessary to develop other specific implementations of the generalized column generation scheme for other classes of bi-objective combinatorial optimization problems.

References

1. 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**(1), 39–50 (2009)
2. Boland, N., Dethridge, J., Dumitrescu, I.: Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters **34**(1), 58–68 (2006)
3. Current, J.R., Schilling, D.A.: The median tour and maximal covering tour problems: Formulations and heuristics. European Journal of Operational Research **73**(1), 114–126 (1994)
4. Das, I., Dennis, J.: A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Structural optimization **14**(1), 63–69 (1997)
5. Delort, C., Spanjaard, O.: Using bound sets in multiobjective optimization: Application to the biobjective binary knapsack problem. In: Experimental Algorithms, pp. 253–265. Springer (2010)
6. Desaulniers, G., Desrosiers, J., Solomon, M.M.: Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. In: Essays and Surveys in Metaheuristics, *Operations Research/Computer Science Interfaces Series*, vol. 15, pp. 309–324. Springer US (2002)
7. Ehrgott, M., Gandibleux, X.: Bound sets for biobjective combinatorial optimization problems. Computers & Operations Research **34**(9), 2674–2694 (2007)
8. Ehrgott, M., Tind, J.: Column generation in integer programming with applications in multicriteria optimization. Tech. rep., Faculty of Engineering, University of Auckland, New Zealand. (2007)
9. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to some vehicle routing problems. Networks **44**(3), 216–229 (2004)

10. Fishburn, P.C.: Additive Utilities with Incomplete Product Sets: Application to Priorities and Assignments. Operations Research **15**(3), 537–542 (1967)
11. Gendreau, M., Laporte, G., Semet, F.: The Covering Tour Problem. Operations Research **45**(4), 568–576 (1997)
12. Hà, M.H., Bostel, N., Langevin, A., Rousseau, L.M.: An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. European Journal of Operational Research **226**, 211–220 (2013)
13. Hachicha, M., Hodgson, M.J., Laporte, G., Semet, F.: Heuristics for the multi-vehicle covering tour problem. Computers & Operations Research **27**(1), 29–42 (2000)
14. Haimes, Y.Y., Lasdon, L.S., Wismer, D.A.: 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)
15. Hodgson, M.J., Laporte, G., Semet, F.: A Covering Tour Model for Planning Mobile Health Care Facilities in SuhumDistrict, Ghama. Journal of Regional Science **38**(4), 621–638 (1998)
16. Jozefowiez, N.: A Branch-and-Price Algorithm for the Multi-Vehicle Covering Tour Problem. Rapport LAAS 12686, LAAS-CNRS, France (2012)
17. Jozefowiez, N., Semet, F., Talbi, E.G.: The bi-objective covering tour problem. Computers & Operations Research **34**(7), 1929–1942 (2007)
18. Labbé, M., Laporte, G.: Maximizing User Convenience and Postal Service Efficiency in Post Box Location. Cahiers du GÉRAD. CIRRELT (1986)
19. Peng, F., Jia, X., Gu, X., Epelman, M.A., Romeijn, H.E., Jiang, S.B.: A new column-generation-based algorithm for VMAT treatment plan optimization. Physics in Medicine and Biology **57**(14), 4569–4588 (2012)
20. Reinhardt, L.B., Pisinger, D.: Multi-objective and multi-constrained non-additive shortest path problems. Computers & Operations Research **38**(3), 605–616 (2011)
21. Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks **51**(3), 155–170 (2008)
22. Salari, E., Unkelbach, J.: A column-generation-based method for multi-criteria direct aperture optimization. Physics in medicine and biology **58**(3), 621–639 (2013)
23. Sarpong, B.M., Artigues, C., Jozefowiez, N.: Column Generation for Bi-Objective Vehicle Routing Problems with a Min-Max Objective. In: 13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, *OASICS*, vol. 33, pp. 137–149. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2013)
24. Sarpong, B.M., Artigues, C., Jozefowiez, N.: Using Column Generation to Compute Lower Bound Sets for Bi-Objective Combinatorial Optimization Problems. Rapport LAAS 13437, LAAS-CNRS, France (2013)
25. Sourd, F., Spanjaard, O.: A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. INFORMS Journal on Computing **20**(3), 472–484 (2008)
26. Villarreal, B., Karwan, M.H.: Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. Mathematical Programming **21**(1), 204–223 (1981)