A New Formulation and a Branch-and-Cut Algorithm for the Set Orienteering Problem

C. Archetti[2], F. Carrabs*[1], R. Cerulli[1], and F. Laureana[1]

[1] *Department of Mathematics, University of Salerno, Italy,*
*flaureana@unisa.it, fcarrabs@unisa.it, raffaele@unisa.it, Corresponding author**
[2] *Department of Information Systems, Decision Sciences and Statistics, ESSEC Business School in Paris, France,*
archetti@essec.edu

Abstract

In this study we address the Set Orienteering Problem, which is a generalization of the Orienteering Problem where customers are clustered in groups. Each group is associated with a profit which is gained in case at least one customer in the group is served. A single vehicle is available to serve the customers. The aim is to find the vehicle route that maximizes the profit collected without exceeding a maximum route cost, which can be interpreted also as route duration. The problem was introduced in [2] together with a mathematical programming formulation. In this paper, we propose a new formulation which uses less variables. We also derive different classes of valid inequalities to strengthen the formulation. In addition, separation algorithms are developed, some of which are new with respect to those presented in the literature. A branch-and-cut algorithm is implemented to solve the problem and tests are made on benchmark instances. The results show that the branch-and-cut algorithm is effective in solving instances with up to 100 customers. Moreover, the difficulty of solving the problem largely depends on the maximum route duration. We also show that valid inequalities are effective in speeding up the solution process. Finally, a comparison with two exact benchmark approaches proposed in the literature shows that the branch-and-cut algorithm proposed in this paper is the new state-of-the-art exact approach for solving the Set Orienteering Problem.

Keywords: Routing; Orienteering Problem; Integer Linear Programming; Branch-and-cut.

1 Introduction

The class of routing problems with profits have received a lot of attention in recent years thanks to their many practical applications. We refer the reader to the surveys by [3] and [16] for an exhaustive review of the literature on these problems. The most widely known problem in the class is the Orienteering Problem (OP) which was introduced in [24]. In the OP, there is a single vehicle, leaving and returning to a depot, which serves a set of geographically dispersed customers. Each customer is associated with a profit which is collected in case the customer is served. The aim is to define the vehicle route maximizing the collected profit in such a way that the route cost does not exceed a maximum threshold.

The literature on OP is wide and the problem has recently found interesting applications in the context of last-mile (or same-day) delivery services (see [26] and [17]). In addition, generalizations of the problems also emerged considering the cases in which customers are clustered and profits are associated with cluster of customers. A first contribution related to such a case is by [1] where the Clustered Orienteering Problem (COP) is studied, in which the profit of a cluster is collected in case all customers in the cluster are visited. On the opposite, in [2] the Set Orienteering Problem (SOP) was introduced, where the profit of a cluster is collected in case at least one customer from the cluster is served.

In the current paper, we focus on the SOP. As mentioned in [2], the SOP finds application in mass distribution products where customers are clustered in groups and carriers stipulate the contract with each group. Once the contract is signed, the carrier receives the corresponding compensation in case the delivery is performed to one customer from the group, which is chosen by the carrier. The entire quantity requested by the group is delivered to that single customer, which will then distribute it among the others. This allows the carrier to offer a better price for delivery. Another application arises when private customers group together either to reach the minimum quantity order required by the carrier or to reach large quantity orders to hopefully gain a lower price. Typically, in this case, the delivery is made to a single location. Finally, SOP finds application in the recent domain of last-mile fast delivery services. Specifically, when the service to a customer can be made in different locations (for example, customer's place, pickup station, delivery locker), then the carrier can choose what is the most convenient place to deliver the order. This problem has been recently studied in the literature under the name 'vehicle routing problem with delivery options' (see [11, 12, 18, 23, 25, 27]). The SOP models the case where a single vehicle is available to perform the deliveries and a prize is associated with each customer and collected only in case the delivery is performed. In this case, each customer represents a cluster and the nodes in each cluster represent the locations where the delivery can be performed.

Heuristic algorithms for the SOP have been proposed in [20] and [5]. In the first paper, the authors propose a Variable Neighborhood Search (VNS) which is applied, in addition to the SOP, also to the Orienteering Problem with Neighborhoods (OPN) and the Dubins Orienteering Problem (DOP). The OPN is the problem in which the profit of a customer is collected within a radius centered to the customer itself, while the DOP uses airplane-like smooth trajectories to connect individual customers (see [4], [22], [21]). To the best of our knowledge, the only contributions proposing a mathematical formulation for the SOP are [2] and [20]. In both cases, the model includes binary variables associated with a visit to each customer. The difference among the two formulations is that, in [2], subtour elimination constraints are formulated in a polynomial way using the Miller-Tucker-Zemlin (MTZ) formulation (see [9]). Instead, in [20] subtour elimination constraints are formulated as exponentially many connectivity constraints.

The contribution of the current work can be summarized as follows:

- We propose a new mathematical formulation for the SOP which does not include binary visiting variables associated with customers. Also, subtour elimination constraints are formulated through exponentially many connectivity constraints linking arc-flow variables with binary variables associated with clusters visit.

- We introduce different classes of valid inequalities to strengthen the formulation. Some of them are inherited from existing works while others are new and tailored to the new formulation.

- We propose separation algorithms for subtour elimination constraints and valid inequalities. Specifically, we propose a novel algorithm for a class of valid inequalities called path inequalities.

- We perform extensive computational tests on benchmark instances. The novel formulation, with the addition of valid inequalities, is solved through a branch-and-cut algorithm. The results show that the approach is effective in handling instances with up to 100 customers and that the problem difficulty largely depends on the maximum route duration. The formulation, and the corresponding branch-and-cut algorithm, performs favorably against exact approaches proposed in the literature both in terms of computational time and number of instances solved to optimality.

The paper is organized as follows. In Section 2 we formally describe the problem, present a mathematical formulation proposed in the literature and propose a new formulation. We also show that that new formulation has a stronger relaxation than the benchmark formulation. Section 3 presents the valid inequalities used to strengthen the formulation while in Section 4 we describe the branch-and-cut algorithm. Computational results are presented in Section 5. Finally, in Section 6 some conclusions are drawn.

2 Problem description and mathematical formulations

The SOP can be described as follows. Let $G = (V, A)$ be a complete directed graph, where $V = \{0\} \bigcup C$. Vertex 0 is the depot where the route of the vehicle starts and ends, while C is the set of customers, and it is partitioned into l clusters, $C_1, ..., C_l$. Let us denote by $\mathcal{P} = \{0, 1, ..., l\}$ the set of the indices associated with clusters. Each cluster C_g has an associated profit p_g, which is collected when at least a customer $i \in C_g$ is visited. The profit of a cluster can be collected at most once. Cluster 0 corresponds to the cluster containing the depot only and is associated with a profit $p_0 = 0$. Furthermore, let c_{ij} be a non-negative cost associated with arc $(i, j) \in A$. The *Set Orienteering Problem (SOP)* consists in finding a route maximizing the collected profit, assuring that the associated cost does not exceed a maximum value T_{max}. In what follows, we assume that the costs c_{ij} satisfy the triangle inequality. This implies that there exists an optimal solution containing at most one vertex for each visited cluster.

In [2] and [20], the authors proposed a 0-1 Integer Linear Program (ILP) for the SOP using the three following classes of binary variables:

- y_i, $\forall i \in V$, equals to 1 if vertex i is visited, 0 otherwise;
- x_{ij}, $\forall (i,j) \in A$, equals to 1 if arc (i,j) is traversed, 0 otherwise;
- z_g, $\forall g \in \mathcal{P}$, equals to 1 if at least a vertex in cluster C_g is visited, 0 otherwise.

Given $A' \subseteq A$ and $V' \subseteq V$, we use the notations $x(A') = \sum_{(i,j)\in A'} x_{ij}$, and $y(V') = \sum_{v \in V'} y_v$. Furthermore, for $\mathcal{Q} \subseteq \mathcal{P}$, we use the notation $z(\mathcal{Q}) = \sum_{g \in \mathcal{Q}} z_g$. For $S, S' \subseteq V$, we denote by $A(S : S') = \{(u,v) \in A : u \in S, v \in S'\}$ the set of arcs having source in S and sink in S', and by $A(S) = A(S : S)$ the set of the arcs having both extremes in S. Given a subset $S \subseteq V$, we denote by $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ the set of its outgoing arcs, and by $\delta^-(S) = \{(i,j) \in A : i \notin S\, j \in S\}$ the set of its incoming arcs. When $S = \{v\}$, $\delta^+(\{v\})$ and $\delta^-(\{v\})$ are substituted by $\delta^+(v)$ and $\delta^-(v)$ for the ease of reading.

In this formulation, connectivity is ensured by *cutset constraints* expressed by using vertices visiting variables y, thus we call the formulation *cutset formulation (cut)*. It reads as follows:

$$\textbf{(cut)} \quad \text{Maximize } z = \sum_{g \in \mathcal{P}} p_g z_g \tag{1}$$

subject to

$$y_0 = 1 \tag{2}$$

$$\sum_{(i,j)\in A} c_{ij}x_{ij} \le T_{max} \tag{3}$$

$$x(\delta^+(i)) = y_i \qquad i \in V \tag{4}$$

$$x(\delta^-(i)) = y_i \qquad i \in V \tag{5}$$

$$x(\delta^+(S)) \ge y_h \qquad S \subset V, 0 \in S, h \notin S \tag{6}$$

$$y(C_g) = z_g \qquad g \in \mathcal{P} \tag{7}$$

$$x_a \in \{0,1\} \qquad a \in A \tag{8}$$

$$y_v \in \{0,1\} \qquad v \in V \tag{9}$$

$$z_g \in \{0,1\} \qquad g \in \mathcal{P} \tag{10}$$

The objective function (1) maximizes the collected profit. Constraint (2) ensures that the depot belongs to the tour. Constraint (3) guarantees that the total cost of the tour is lower than T_{max}. Constraints (4) and (5) ensure that exactly one arc enters and leaves every visited node. Constraints (6) avoid subtours among visited nodes. Finally, constraints (7) ensure that variable z_g is equal to 0 when no customer in cluster C_g is visited. Constraints (8)–(10) define the variables domain.

We note that in [2] subtour elimination constraints are formulated as MTZ constraints (see [9]), while in [20] they are formulated as:

$$x(A(S)) \le y(S\backslash\{h\}) \qquad S \subset V, 0 \in S, h \in S.$$

The first main contribution of this paper is the proposal of a new formulation for the SOP that does not include vertex visiting variables y. Indeed, as the profit is associated with clusters of customers, the value of a solution is measured through z variables. Also, connectivity constraints can be modelled by using z variables. Note that both formulations proposed in [2] and [20] use vertices visiting variables. A first clear advantage of our new formulation is related to the fact that it uses fewer binary variables. Also, we show in the following that the new formulation provides a stronger lower bound associated with the linear relaxation, as the polyhedron of the new formulation is contained in the one of the **cut** formulation.

Our formulation uses only two families of binary variables, namely, x_{ij} and z_g. Connectivity is ensured by *cluster cutset constraints*, thus we call it *cluster cutset formulation (clucut)*. It reads as follows:

$$\textbf{(clucut)} \qquad \text{Maximize } z = \sum_{g \in \mathcal{P}} p_g z_g \tag{11}$$

subject to

$$z_0 = 1 \tag{12}$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \leq T_{max} \tag{13}$$

$$x(\delta^+(i)) = x(\delta^-(i)) \qquad i \in V \tag{14}$$

$$x(\delta^+(S)) \geq z_g \qquad g \in \mathcal{P}, S \subset V \setminus C_g, 0 \in S \tag{15}$$

$$x(\delta^+(C_g)) = z_g \qquad g \in \mathcal{P} \tag{16}$$

$$x(\delta^-(C_g)) = z_g \qquad g \in \mathcal{P} \tag{17}$$

$$x(A(C_g)) = 0 \qquad g \in \mathcal{P} \tag{18}$$

$$x_a \in \{0, 1\} \qquad a \in A \tag{19}$$

$$z_g \in \{0, 1\} \qquad g \in \mathcal{P} \tag{20}$$

The objective function (11) corresponds to maximizing the profit collected, as in the **cut** formulation. Constraint (12) ensures that the cluster containing the depot belongs to the tour. Constraint (13) is the same as constraint (3) and ensure that the maximum duration T_{max} is not exceeded. Constraints (14) are flow balance constraints associated with each node.

Constraints (15) avoid subtours among visited clusters. Constraints (16) and (17) ensure that variable z_g is equal to 1 if and only if at least one vertex in cluster C_g is visited. Constraints (18) ensure that intra-cluster arcs are equal to zero. Note that (18) are not needed for the correctness of the formulation and are used to strengthen it. Finally, (19) and (20) are variables domain.

2.1 Properties of the cluster cutset formulation

We first show that formulation **clucut** is a valid formulation for the SOP by showing that any feasible solution of **clucut** can be transformed in a feasible solution of **cut** and vice versa.

Theorem 1. *Any feasible solution of **cut** can be transformed in a feasible solution of **clucut** and vice versa.*

Proof. Let us consider a feasible solution of **cut**. Note that, because of (7), for each cluster $g \in \mathcal{P}$ we have that $y(C_g) \leq 1$. Let $\mathcal{P}' \subseteq \mathcal{P}$ be the subset of clusters visited. A feasible solution for **clucut** is obtained by simply setting the values of the variables x and z equal to the corresponding value in the feasible solution of **cut**. Note that this solution is feasible for **clucut**. In fact, constraints (12)–(14) correspond to (2)–(5). Also, by combing constraints (4) (or (5)) and (7), we obtain (16) (or (17)) and (18). Similarly, by combining (7) and (6), we obtain (15). Thus, any feasible solution of **cut** can be transformed in a feasible solution for **clucut**.

Let us now consider a feasible solution for **clucut**. Similarly as above, let $\mathcal{P}' \subseteq \mathcal{P}$ be the subset of clusters visited. Because of constraints (16), (17) and (14), there is at most one vertex visited for each cluster $g \in \mathcal{P}'$. Let us call this vertex i_g. A solution of **cut** is obtained by setting the values of the variables x and z equal to the corresponding value in the feasible solution of **clucut** and by setting $y_{i_g} = 1$, $g \in \mathcal{P}'$. This way, constraints (2)–(5) and (7) are satisfied. Also, because of (15), connectivity is guaranteed so (6) are satisfied as well. □

We now show that formulation **clucut** provides a strongest relaxation than the one associated with formulation **cut** as the polyhedron of **clucut** is included in the one of **cut**.

Let us denote by P_{cut} and P_{clucut} the polytopes associated with the linear relaxation of formulations *cut* and *clucut*, respectively, projected in the space of x and z variables.

The following proposition holds.

Theorem 2. $P_{clucut} \subset P_{cut}$.

Proof. First, we show that, given $(x, z) \in P_{clucut}$, there exists a vector y such that $(x, y, z) \in P_{cut}$. Let us choose y in such a way that $y_i = x(\delta^+(i)) = x(\delta^-(i))$, for any $i \in V$. Because of constraints (16), (17) and (7), (x, y, z) satisfies constraints (4) and (5). Constraint (2) is implied by (12). Therefore, to prove that (x, y, z) belongs to P_{cut}, it is sufficient to show that it satisfies constraints (6). Indeed, by combining (15) with (16) or (17), we obtain (6). This proves that $P_{clucut} \subseteq P_{cut}$.

As for the strict inclusion, the solution depicted in Figure 1 shows a solution which is feasible for the linear relaxation of *cut*, but not of the one of *clucut*. Vertex 0 corresponds to the depot and the 5 customers are grouped in 4 clusters. The numbers close to each vertex and arc show the value of

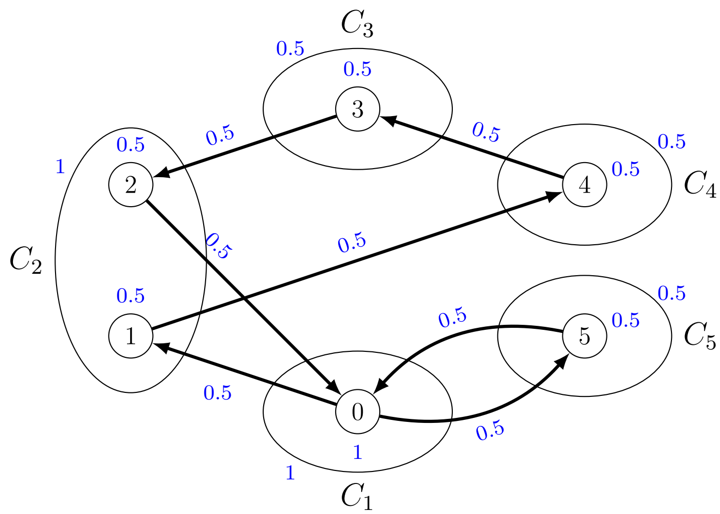

Figure 1: A solution in $P_{cut} \setminus P_{clucut}$

the y and x variables, respectively, in the solution of the linear relaxation of **cut**. We can notice that all constraints of **cut** are satisfies while constraints (15) are violated for $S = \{0, 5\}$ and $C_g = C_2 = \{1, 2\}$. □

3 Valid Inequalities

In the following we describe the valid inequalities we used to strengthen formulation **clucut**. They are inherited from the literature about routing problems in general and are adapted to the SOP.

Given an instance of the SOP defined on graph G, let us consider the following polytope:

$$P(G) \quad = conv\{(x, z) \in \mathbb{R}^{m+n} : (x, z) \text{ satisfies } (12) - (20)\}.$$

3.1 Cover Inequalities

Cover inequalities enhance T_{max} constraint (13). Specifically, given $T \subset A$, such that $\sum_{(i,j)\in T} c_{ij} > T_{max}$, we have that

$$x(T) \leq |T| - 1 \tag{21}$$

is valid for $P(G)$. Inequality (21) can be further strengthened by considering all the arcs of the graph having cost at least equal to $max_{(i,j)\in T} c_{ij}$ (see [14]). In such a way we obtain the following inequality, which is valid for $P(G)$:

$$x(T \cup \bar{T}) \leq |T| - 1, \qquad (22)$$

where $\bar{T} = \{(h,k) \in A \setminus T : c_{hk} \geq max_{(i,j)\in T} c_{ij}\}$.

3.2 Conditional Cuts

Fischetti et al. in [14] introduced the so-called conditional cuts for the OP. In the current paper, we adapted them to the SOP. Let z^{LB} be a lower bound on the optimal solution to the SOP. Given a subset of nodes, $S \subset V \cup \{0\}$, such that $0 \in S$, we denote by $\mathcal{P}_S$ the subset of $\mathcal{P}$ such that $\mathcal{P}_S = \{g \in \mathcal{P} : S \cap C_g \neq \emptyset\}$. If it results that $\sum_{g\in\mathcal{P}_S} p_g \leq z^{LB}$, then the following inequality

$$x(A(S)) \leq z(\mathcal{P}_S) - 1, \qquad (23)$$

is satisfied by the optimal solution to the SOP.

3.3 Cluster Cover Inequalities

Let z^{UB} be an upper bound on the optimal solution to the SOP. Given a subset $\mathcal{Q} \subset \mathcal{P}$, such that $\sum_{g\in\mathcal{Q}} p_g > z^{UB}$, the following inequality

$$z(\mathcal{Q}) \leq |\mathcal{Q}| - 1, \qquad (24)$$

is satisfied by the optimal solution to the SOP. These inequalities are adapted from the vertex cover inequalities, which were first proposed for the OP in [15].

3.4 Path Inequalities

Path inequalities were first introduced in [14] for the Orienteering Problem and we adapt them to the SOP. The idea is the following. Given k clusters, $C_{i_1}, ..., C_{i_k}$, let $I = \{(i_1, i_2), ..., (i_{k-1}, i_k)\}$ be a simple path through nodes $V(I) = \{i_1, ..., i_k\} \subset C$, such that $i_1 \in C_{i_1}, ..., i_k \in C_{i_k}$. We define the following subset of nodes:

$$W(I) = \{v \in V \setminus V(I) : I \cup (i_k, v) \text{ is part of a feasible SOP solution}\}.$$

$W(I)$ represent the set of all vertices to which path I can be feasibly extended. Path inequalities are formulated as:

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} - \sum_{j=2}^{k-1} z_{i_j} - \sum_{v \in W(I)} x_{i_k v} \leq 0. \tag{25}$$

Theorem 3. *Inequalities* (25) *are valid for* $P(G)$.

Proof. Let us suppose that there exists a feasible solution to the SOP, (x^*, z^*), violating (25). This implies that

$$x^*_{i_1 i_2} + (x^*_{i_2 i_3} - z^*_{i_2}) + ... + (x^*_{i_{k-1} i_k} - z^*_{i_{k-1}}) - \sum_{v \in W(I)} x_{i_k v} > 0.$$

Let us note that, thanks to constraints (16), it results that $x^*_{i_j i_{j+1}} - z^*_{i_j} \leq 0$, for $j = 2, ..., k-1$. Therefore, for the left hand side to be greater than zero, it must be $x^*_{i_1 i_2} = 1$, $x^*_{i_j i_{j+1}} - z^*_{i_j} = 0$, for any $j = 2, ..., k-1$, and $x^*_{i_k v} = 0$, for any $v \in W(I)$. It follows that $x^*_{i_j i_{j+1}} = 1$, for any $j = 1, ..., k-1$, and $z^*_{i_1} = z^*_{i_2} = ... = z^*_{i_k} = 1$, and this means that (x^*, z^*) contains every arc in I, and at least an arc (i_k, v), with $v \notin W(I)$, which cannot be according to the definition of $W(I)$. □

4 Branch-and-cut algorithm

We devised a branch-and-cut algorithm for the SOP based on the *clucut* formulation. A preprocessing procedure is carried out before executing the algorithm by using the properties introduced in [5]. This procedure is aimed at removing from G the vertices, arcs and clusters that are not necessary to build an optimal solution. In particular, for the edge removal, we use the procedure proposed in [7] and successfully applied also in [6, 8].

The initial LP model is obtained by removing constraints (15), and relaxing the integrality constraints on the variables of the original formulation.

For any subproblem $\mathcal{L}$ of the branch-and-bound tree, we compute the optimal Linear Programming (LP) solution $(x^*_{LP}(\mathcal{L}))$. If $x^*_{LP}(\mathcal{L})$ is not feasible, we search for violated constraints (15), (22), (23), (24) and (25). When no violation is identified, the algorithm performs the branching on a fractional variable, by using the default CPLEX parameters.

As for the sequence in which subtour elimination constraints (15) and valid inequalities (22)–(25) are separated in each node of the branch-and-bound tree, after some preliminary experiments, we determined that the most efficient setting is the following. First, cover inequalities (22) and cluster cover inequalities (24) are separated. Subtour elimination constraints (15) are separated after (22) and (24). Moreover, except for the root node, they are separated only in case no violated inequality (22) and (24) is identified. Set S determined by the separation algorithm of (15) is checked also for potential violation of conditional cuts (23). Lastly, path inequalities (25), whose separation procedure is time consuming, are separated only in case none of the former inequalities is violated.

In the following subsection, we describe the separation procedures used for each family of valid inequalities. While most procedures are inherited from benchmark algorithms proposed in the literature, we propose a new procedure for separating path inequalities.

4.1 Separation Procedures

In the following we denote by $G^* = (V^*, A^*)$ the graph associated with the fractional solution (x^*, z^*). More in detail, $V^* = \{i \in V : \exists (i,j) \in \delta^+(i) : x^*_{ij} > 0\} \cup \{i \in V : \exists (j,i) \in \delta^-(i) : x^*_{ji} > 0\}$ and $A^* = \{(i,j) \in A : x^*_{ij} > 0\}$.

Subtour elimination constraints (15) are separated through the classical exact in polynomial time min-cut algorithm (see [19]).

Cover inequalities (22) are separated by using the following exact separation procedure. We determine a set $T \subseteq A$ such that $\sum_{(i,j)\in T} c_{ij} > T_{max}$ and for which $x^*(T) - |T| + \epsilon$ is maximum by solving the following Knapsack Problem:

$$\text{Minimize } \tilde{z} = \sum_{(i,j)\in A} (1 - x^*_{ij})\tilde{z}_{ij}$$

$$\sum_{(i,j)\in A} c_{ij}\tilde{z}_{ij} \geq T_{max} + \epsilon$$

$$\tilde{z}_{ij} \in \{0,1\}, (i,j) \in A$$

If the optimal solution is strictly less than 1, inequality (21) corresponding to subset $T = \{(i,j) \in A : \tilde{z}^*_{ij} = 1\}$ is violated. Furthermore, to obtain the stronger inequality (22), we simply look for any arc with cost greater than $max_{(i,j)\in T} c_{ij}$, and we add it to T. Note that a similar procedure was proposed in [14] for the OP. Also note that, for the results presented in Section 5.3, the value of ϵ is set to 1 as distances are integer in the tested instances.

Conditional cuts (23) are separated heuristically by using a procedure based on the separation algorithm for constraints (15). Indeed, given z^{LB}, the value of the best integer solution found so far, and S^*, a subset obtained by computing the minimum cut on $\tilde{G}$, if it results that $\sum_{g \in \mathcal{P}_{S^*}} p_g \leq z^{LB}$, then we check if $x(A(S^*)) > z(\mathcal{P}_{S^*}) - 1$, and if so we add the corresponding violated inequality (23) to the formulation.

Cluster cover inequalities (24) are separated exactly by solving the following knapsack problem:

$$\text{Minimize } u = \sum_{g \in \mathcal{P}} (1 - z_g^*) u_g$$

$$\sum_{g \in \mathcal{P}} p_g u_g \geq z^{UB} + 1$$

$$u_g \in \{0, 1\}, g \in \mathcal{P}$$

where z^{UB} is the value of the current upper bound. It is easy to see that if the optimal solution is strictly less than 1, by choosing $\mathcal{Q} = \{g \in \mathcal{P} : u_g^* = 1\}$, we obtain a violated inequality (24).

Path inequalities (25) are separated heuristically by using the following procedure. Given the graph $G^* = (V^*, A^*)$ we define a weight function, $w : A^* \to \mathbb{R}$, such that $w_{ij} = z_i^* - x_{ij}^*$, for any $(i, j) \in A^*$. For any $u, v \in V \setminus \{0\}$ such that u and v belong to different clusters, we compute a path $I = \{(i_1, i_2), ..., (i_{k-1}, i_k)\}$ from $u = i_1$ to $v = i_k$ in G^*, assuring that it contains at most one vertex for each cluster and having minimum weight, $w(I) = \sum_{j=1}^{k-1} w_{i_j i_{j+1}}$. It is easy to see that

$$w(I) = \sum_{j=1}^{k-1} z_{i_j}^* - \sum_{j=1}^{k-1} x_{i_j i_{j+1}}^*.$$

To compute I for a given pair of vertices u, v we solve the following ILP where s_{ij} is a binary variables taking value 1 if arc (i, j) is traversed and t_i is a binary variables equal to 1 if vertex i is visited:

$$\text{Minimize } w = \sum_{(i,j)\in A^*} w_{ij} s_{ij} \tag{26}$$

$$\sum_{(i,j)\in\delta^+(i)} s_{ij} - \sum_{(l,i)\in\delta^-(i)} s_{li} = \begin{cases} 1, & i = u \\ 0, & i \neq u, v \quad i \in V^* \\ -1, & i = v \end{cases} \tag{27}$$

$$\sum_{i\in C_g\cap V^*} t_i \leq 1, \quad g \in \mathcal{P} \tag{28}$$

$$t_i \geq s_{ij}, \quad (i,j) \in A^* \tag{29}$$

$$t_j \geq s_{ij}, \quad (i,j) \in A^* \tag{30}$$

$$t_i \leq \sum_{j\in FS(i)} s_{ij} + \sum_{l\in BS(i)} s_{li}, \quad i \in V^* \tag{31}$$

$$t_u = 1 \tag{32}$$

$$t_v = 1 \tag{33}$$

$$s_{ij} \in \{0,1\}, \quad (i,j) \in A^* \tag{34}$$

$$t_i \in \{0,1\}, \quad i \in V^* \tag{35}$$

Formulation (26)–(35) aims at finding the shortest path from u to v in G^*. The objective function (26) minimizes the value of $w(I)$. Constraints (27) are flow conservation constraints while (28) establish that one vertex per cluster at most is visited. Constraints (29) and (30) force t variables to take value 1 in case an arc incident to the corresponding vertex is traversed. Constraints (31) fix the value of variable t_i to 0 in case vertex i is not visited. (32) and (33) force the visit of vertices u and v, respectively, while (34)–(35) define variables domain. Note that constraints (31)–(33) are not needed. However, they strengthen the formulation and speed up the solution process.

If $w(I) - z^*_{i_1} \geq 0$, then I does not lead to a violated inequality (25). Otherwise, we build $W(I)$ as follows: for any $v \in V \setminus V(I)$, if $c_{0i_1} + \sum_{(i,j)\in I} c_{ij} + c_{i_k v} + c_{v0}$ is lower than T_{max}, then we add v to $W(I)$. Finally, we check if $-w(I) + z^*_{i_1} - \sum_{v\in W(I)} x^*_{i_k v}$ is strictly greater than 0, and in such a case we have identified a violated inequality (25). The idea is that first a new vertex v is added to the path and then the inequality is checked for violation.

Note that path inequalities were introduced in [14]. However, the separation procedure used in the latter paper was different and it consisted in an enumeration scheme to detect path I. This scheme is not suitable for the SOP as a further check is needed to assure that all nodes in I belong to different clusters, making the procedure too slow to be applied in an effective way. This has motivated the design of the new procedure described above.

5 Computational Tests

In this section, we present the results obtained by the branch-and-cut algorithm described in Section 4. The branch-and-cut algorithm (called 'BC' from now on) was coded in C++ using the LEMON graph library [10]. All tests were performed on an OSX platform (iMac 2020), running on an Intel Core i9-10910 processor clocked at 3.6 GHz with 64 GB of RAM. The mathematical formulation was solved using the ILOG Concert Technology library and CPLEX 20.1 in single thread mode. A time limit of one hour and a memory limit of 5GB were imposed. All remaining CPLEX parameters were left to their default value.

The computational tests were carried out on the instances proposed in [2] and named *Set1*. This set of instances was obtained by adapting the instances of the Generalized Traveling Salesman Problem proposed in [13]. The number of vertices ranges from 52 to 1084 but we only consider the instances with up to 198 nodes for our computational tests as for higher dimensions the exact approach is not practicable. The number of clusters is equal to $\sim 20\%$ the number of vertices. T_{max} is set to $\omega \times GTSP^*$, where $GTSP^*$ is the best-known solution value of the GTSP (taken from [13]) and ω is set to 0.4, 0.6, and 0.8. Finally, two rules, g_1 and g_2, are used to assign the profit to the clusters. The first rule sets the profit of each cluster C_g equal to $|C_g|$. The second rule assigns to vertex j a profit equal to $1 + (7141j + 73) mod(100)$ and the profit of a cluster is given by the sum of the values of the profit of all vertices belonging to it.

Another set of instances for SOP, named *Set2*, was introduced in [2]. The instances in this set are the same as those in *Set1*. More specifically, the instances contain the same vertices and the same number of clusters as the instances in *Set1*, but the vertices are assigned in a random way to the clusters. Both sets contain 27 instances each. The datasets and the tables of the paper are available here `https://github.com/fcarrabs/Set_Orienteering_Problem`

The section is organized as follows. We first analyze the effectiveness of

	C-BC	NoCond		NoCover		NoCluCover		NoPath	
	Time	Time	Gap%	Time	Gap%	Time	Gap%	Time	Gap%
				$\omega = 0.4$ and g_1					
AVG	622.50	610.86	0.80%	1229.98	-2.73%	615.23	0.00%	1361.12	-8.08%
#Worse			0		7		0		11
				$\omega = 0.4$ and g_2					
AVG	808.27	696.90	0.31%	1230.15	-2.89%	751.66	0.00%	1479.63	-8.06%
#Worse			0		7		0		13
				$\omega = 0.6$ and g_1					
AVG	934.94	1004.03	0.13%	1573.66	-1.83%	932.39	0.00%	1765.57	-2.43%
#Worse			1		5		0		6
				$\omega = 0.6$ and g_2					
AVG	1208.51	1104.96	-0.15%	1653.26	-0.28%	1188.54	0.00%	1580.25	-0.66%
#Worse			1		2		0		2
				$\omega = 0.8$ and g_1					
AVG	909.19	1222.59	-0.83%	2174.25	-4.99%	908.76	0.00%	2130.63	-4.90%
#Worse			2		10		0		10
				$\omega = 0.8$ and g_2					
AVG	1768.56	1744.82	0.09%	2290.29	-2.98%	1845.61	-0.08%	2362.75	-3.30%
#Worse			1		8		1		8

Table 1: Summary of the computational results for the five versions of the branch-and-cut algorithm on all instances.

the valid inequalities presented in Section 3. Specifically, in Section 5.1, we present the results of tests in which we run the branch-and-cut algorithm with the full set of valid inequalities and we compare it with the version of the algorithm in which we discard one inequality at a time. These tests enable us to determine the best version of the algorithm which is used to run the final tests. In Section 5.2 we provide the detailed results of the final version of the algorithm and we provide some statistics on the performance of the algorithm. Finally, in Section 5.3 we compare the performance of our algorithm with two benchmark approaches available in the literature.

5.1 Valid Inequalities Effectiveness

As mentioned above, this section is dedicated to evaluate the effectiveness of the valid inequalities presented in Section 3. Specifically, we run the branch-and-cut algorithm with the full set of inequalities and we compare it with the versions in which we discard a single inequality at a time. Tests are made over all instances presented above.

The results of this comparison are summarized in Table 1 for the instances with $\omega = 0.4, 0.6, 0.8$ and profits g_1 and g_2. The detailed results are reported in Tables 12–15 of Appendix A. The first column of the table reports the computational time, in seconds, of the branch-and-cut version with the full set of inequalities (*C-BC*). Four groups of two columns follow, corresponding to the four versions of the branch-and-cut algorithm in which one inequality is excluded: no conditional cuts (*NoCond*), no cover inequalities (*NoCover*), no cluster cover inequalities (*NoCluCover*) and no path inequalities (*NoPath*). Under the *Time* and *Gap%* headings, for each branch-and-cut version, we report the corresponding computational time and the percentage gap between the upper bound obtained by C-BC and the upper bound of the version considered, respectively. This gap is calculated as $gap = \frac{UB_{C-BC} - UB_*}{UB_{C-BC}}$, where UB_{C-BC} and UB_* are the upper bound by C-BC and by the version considered, respectively. Note that positive values of the gap mean that the version of the algorithm without the inequality gives a better result than C-BC. Results are aggregated by values of ω and classes of profit. The rows of the table are organized in groups of two lines reporting, for each version of the algorithm in which an inequality is excluded: the average values of time and gap (*AVG*) and the number of times in which the upper bound provided by the version without an inequality is worse than the one provided by C-BC (*#Worse*).

We note that, when $\omega = 0.4$, cover and path inequalities are indeed effective as their exclusion cause both a remarkable increase in computing time and a deterioration in the value of the upper bound.

In more detail, for both g_1 and g_2 profits, we observe that the upper bound of NoCover worsens 7 times while the computational time increases by~52%, at least. For NoPath the upper bound worsens 11 and 13 times, for g_1 and g_2 respectively, and the computational time increases by ~96%, at least. For $\omega = 0.6$, the contribution of these two inequalities is less impactful but still relevant. For $\omega = 0.8$, the #Worse value of the version without path inequalities is similar to the one observed for $\omega = 0.4$ and profit g_1 while it is lower for profit g_2. Instead, for the version without the cover inequalities, the time, Gap%, and #Worse value are worse than the ones observed for $\omega = 0.4$ and $\omega = 0.6$. The effectiveness of conditional cuts is instead more debatable: indeed, they are not effective when $\omega = 0.4$, as their removal improve the upper bound and reduces the computational time. However, when $\omega = 0.6$, the computational time increases when they are removed for the case g_1 and the upper bound slightly deteriorates for the case g_2. Finally, when $\omega = 0.8$, they are effective in g_1 instances and not effective in g_2 instances. Given that the overall difference between the two versions of the algorithm (with or without conditional cuts) is slightly to the advantage of the first version, we decided to retain them in the final version of the branch-and-cut algorithm. Finally, as for cluster cover inequalities, we notice that they are almost never effective: in fact, the impact on the value of the upper bound is null while the computational time reduces when they are discarded (apart the case $\omega = 0.8$ and g_2). For this reason, we decided to remove these inequalities from the final version of the branch-and-cut algorithm.

5.2 Branch-and-cut performance

The results presented above show that the best version of the branch-and-cut algorithm is the one that does not include cluster cover inequalities. This is the version for which we present the results in the remaining of this section, which is from now on called *BC*.
This section is organized as follows. We first present some statistics about cuts and valid inequalities separation. Then, we analyse the performance of the algorithm over all instances of the testbed and compare it with a version of the algorithm where none of the inequalities presented in Section 3 is used. The aim of this comparison is to show that indeed the inequalities pay-off (apart the cluster cover inequalities as mentioned in the former section).

Specifically, Tables 2–5 present statistics about the separation of subtour elimination constraints and the remaining inequalities for the instances with $\omega = 0.4, 0.6, 0.8$, respectively. The first column reports the name of the instance which contains the reference to the number of customers in the instance (last part of the instance' name). In the second we have the number of nodes of the branch-and-bound tree explored at termination (*Nnodes*). Then we report statistics related to the separation of subtour elimination constraints, conditional cuts, cover inequalities and path inequalities, respectively. Specifically, for each family of inequality, we report the number of inequalities separated (*Num*) and the total computational time for separation (*Time*). The rows of the table are divided in two groups associated with the datasets *Set1* and *Set2*, respectively. Finally, the last row of the table (*AVG*) reports the average values of Nnodes, Num and Time.

Focusing first on Table 2, we see that subtour elimination constraints are by far the most widely violated constraints, followed by path inequalities. The remaining inequalities are much less often violated. As for the time for

				$\omega = 0.4$ and g_1							
		Nnodes	**Subtour**		**Conditional**		**Cover**		**Path**		
	Instance		**Num**	**Time**	**Num**	**Time**	**Num**	**Time**	**Num**	**Time**	
Set1	11berlin52	47	148	0.01	3	0.00	5	0.11	26	0.14	
	11eil51	44	163	0.01	0	0.00	4	0.07	18	0.03	
	14st70	60	217	0.03	0	0.00	3	0.11	29	0.10	
	16eil76	144	709	0.17	2	0.00	12	0.43	182	0.78	
	16pr76	126	705	0.20	0	0.00	6	1.04	51	0.26	
	20kroA100	327	2754	1.49	0	0.00	11	4.32	137	0.62	
	20kroB100	179	1125	0.50	2	0.00	17	1.68	94	0.90	
	20kroC100	86	534	0.16	1	0.00	5	0.62	36	0.23	
	20kroD100	127	670	0.18	1	0.00	15	0.61	125	0.82	
	20kroE100	131	743	0.21	1	0.00	3	0.77	24	0.14	
	20rat99	107	433	0.05	2	0.00	3	0.19	50	0.58	
	20rd100	186	1082	0.31	0	0.00	6	0.78	67	0.28	
	21eil101	150	1229	1.27	0	0.00	11	1.14	112	1.85	
	21lin105	243	1212	0.29	3	0.00	6	1.16	61	0.32	
	22pr107	3	9	0.00	0	0.00	0	0.01	0	0.00	
	25pr124	775	4938	3.13	2	0.00	41	9.91	310	6.27	
	26bier127	948	10409	25.12	1	0.00	13	21.28	184	3.50	
	26ch130	812	9358	19.93	1	0.00	48	14.64	223	9.59	
	28pr136	305	2423	2.31	0	0.00	19	2.81	291	4.34	
	29pr144	1669	16353	22.97	3	0.00	24	21.38	462	5.76	
	30ch150	1468	12015	19.23	17	0.00	66	16.30	528	13.48	
	30kroA150	1118	12733	27.13	2	0.00	57	23.60	358	13.11	
	30kroB150	692	7658	20.54	7	0.00	29	17.78	531	29.73	
	31pr152	1094	11838	14.13	4	0.00	12	14.46	305	4.00	
	32u159	969	10084	14.10	1	0.00	23	10.61	125	3.97	
	39rat195	685	8612	13.78	1	0.00	17	6.61	344	10.00	
	40d198	279	1914	1.23	17	0.00	24	1.66	230	6.65	
Set2	11berlin52	53	283	0.05	4	0.00	4	0.19	37	0.35	
	11eil51	116	466	0.07	1	0.00	18	0.29	151	1.57	
	14st70	41	293	0.06	0	0.00	2	0.13	33	0.21	
	16eil76	103	672	0.26	0	0.00	4	0.36	120	1.05	
	16pr76	225	1765	0.80	3	0.00	3	2.38	44	0.64	
	20kroA100	243	2564	2.30	3	0.00	17	5.51	166	10.56	
	20kroB100	277	2645	2.08	1	0.00	15	2.86	47	4.32	
	20kroC100	260	2051	1.48	0	0.00	23	2.61	157	8.39	
	20kroD100	135	1042	0.59	0	0.00	8	1.43	80	3.87	
	20kroE100	87	790	0.47	1	0.00	6	0.76	39	3.39	
	20rat99	56	416	0.10	8	0.00	2	0.16	5	1.26	
	20rd100	234	1993	1.30	1	0.00	6	1.68	209	11.84	
	21eil101	231	2506	2.93	2	0.00	16	2.43	110	5.54	
	21lin105	496	3880	2.22	42	0.00	24	3.06	430	6.83	
	22pr107	281	2789	0.89	18	0.00	21	1.25	191	4.49	
	25pr124	556	6744	7.66	0	0.00	12	13.41	185	46.82	
	26bier127	837	16399	47.56	23	0.00	23	32.75	260	18.22	
	26ch130	955	10080	26.69	3	0.00	60	19.73	329	268.79	
	28pr136	186	2864	4.03	0	0.00	5	2.12	77	4.56	
	29pr144	402	9044	19.02	0	0.00	10	13.22	336	7.16	
	30ch150	544	8392	21.25	0	0.00	23	9.61	228	112.49	
	30kroA150	375	9272	28.64	4	0.00	11	15.80	173	11.96	
	30kroB150	646	12812	42.42	11	0.00	51	33.13	444	85.94	
	31pr152	457	9460	18.32	0	0.00	18	11.15	328	19.43	
	32u159	333	5439	12.63	0	0.00	19	4.82	154	9.81	
	39rat195	746	14727	38.77	0	0.00	29	8.98	154	59.56	
	40d198	478	7665	12.09	0	0.00	11	4.83	158	47.74	
AVG		**409.76**	**4761.50**	**8.95**	**3.63**	**0.00**	**17.06**	**6.83**	**176.81**	**16.19**	

Table 2: Branch-and-cut statistics on the instances with $\omega = 0.4$ and profit g_1.

separation, we notice that, while subtour elimination constraints are separated very efficiently, the computational time for separating path inequalities is, in comparison, much larger. This justifies our choice of separating path inequalities only when no other inequality is violated. As for the other inequalities, we recall that conditional cuts are separated during the separation of subtour elimination constraints, which explains the short computing time of separation for this class. As for the cover inequalities, the separation time is short so this justifies the choice of keeping these inequalities despite the fact that they are violated quite rarely.

Similar considerations can be done for ω equal to 0.4 and profit g_2 (Table 3). However, an interesting observation here is that instances in g_2 are more difficult to solve as witnessed by the larger number of nodes of the branch-and-bound tree. As a consequence, the number of inequalities separated and the computational time required to separate them almost always increases for all the types of inequalities.

The same trend is observed for values of ω equal to 0.6 and 0.8 (reported in Tables 4 and 5, respectively) where again we notice that the instances with profit g_2 are more difficult to solve with respect to the ones with profit g_1.

We now present the results of the comparison between BC and the branch-and-cut algorithm that solves formulation **clucut** only, i.e., with none of the valid inequalities proposed in Section 3. Formulation **clucut** is solved through branch-and-cut by separating the subtour elimination constraints (15). The separation algorithm is the same as the one used in BC.

The results of the comparison between **clucut** and BC are reported in Tables 6–8 for values of ω equal to 0.4, 0.6 and 0.8, respectively.

Each table is organized as follows. The first column reports the name of the instance. Then, two groups of six columns report the results for profits g_1 and g_2, respectively. Each group is composed by two subgroups of three columns, referring to the results by **clucut**, first, and BC, second. Specifically, we report the value of the best solution found at termination, the computational time (in seconds) and the optimality gap at termination (which is 0% in case the instance is solved to proven optimality). The rows of the table are divided in two groups associated with the datasets *Set1* and *Set2*, respectively. The last two rows of each table report the average computational time and the average optimality gap, and the number of instances solved to optimality, respectively.

The values on the AVG line in Table 6, related to instances with $\omega = 0.4$, show that BC is much faster than **clucut**. Indeed, the average computational

	$\omega = 0.4$ and g_2									
	Instance	**Nnodes**	**Subtour Num**	**Subtour Time**	**Conditional Num**	**Conditional Time**	**Cover Num**	**Cover Time**	**Path Num**	**Path Time**
Set1	11berlin52	60	182	0.02	1	0.00	6	0.14	38	0.12
	11eil51	159	332	0.03	0	0.00	21	0.24	152	0.50
	14st70	74	295	0.03	2	0.00	2	0.15	8	0.03
	16eil76	126	547	0.12	18	0.00	14	0.36	112	0.46
	16pr76	139	599	0.16	13	0.00	9	0.79	89	0.26
	20kroA100	402	2517	1.34	1	0.00	20	5.33	173	3.65
	20kroB100	296	1623	0.76	0	0.00	35	2.91	200	3.45
	20kroC100	194	796	0.24	3	0.00	14	1.11	162	1.53
	20kroD100	288	992	0.26	0	0.00	29	1.48	522	3.03
	20kroE100	168	944	0.28	1	0.00	11	1.15	103	0.98
	20rat99	76	278	0.03	2	0.00	4	0.12	34	0.15
	20rd100	217	1295	0.38	1	0.00	2	1.07	53	1.21
	21eil101	229	1816	1.83	2	0.00	6	1.60	72	0.98
	21lin105	229	944	0.23	13	0.00	11	1.01	128	1.14
	22pr107	7	29	0.00	0	0.00	1	0.02	17	0.03
	25pr124	1135	5835	3.89	4	0.00	85	12.45	629	17.94
	26bier127	2083	18913	48.53	1	0.00	57	37.82	504	32.86
	26ch130	1560	13028	27.41	1	0.00	116	21.11	513	20.82
	28pr136	865	7784	7.49	2	0.00	36	8.75	617	4.83
	29pr144	3682	12905	17.26	8	0.00	490	42.63	1889	98.98
	30ch150	1376	10970	17.88	1	0.00	40	14.12	548	33.02
	30kroA150	1473	12756	25.96	1	0.00	111	32.42	898	32.75
	30kroB150	2359	17929	47.76	3	0.00	258	54.98	1335	60.74
	31pr152	1099	11895	14.29	2	0.00	14	14.50	270	4.20
	32u159	884	9275	12.98	1	0.00	15	11.23	139	4.00
	39rat195	749	7499	12.40	4	0.00	81	8.67	920	30.59
	40d198	422	2967	1.84	2	0.00	10	2.45	142	2.17
Set2	11berlin52	36	211	0.03	3	0.00	4	0.14	35	0.47
	11eil51	71	308	0.05	2	0.00	5	0.14	24	0.23
	14st70	78	401	0.07	3	0.00	5	0.18	35	0.20
	16eil76	273	1224	0.47	1	0.00	20	0.95	274	7.02
	16pr76	214	1353	0.55	0	0.00	12	2.19	131	6.31
	20kroA100	342	2832	2.53	1	0.00	35	6.59	237	20.89
	20kroB100	396	2920	2.40	3	0.00	37	6.64	183	40.50
	20kroC100	808	3278	2.48	3	0.00	91	7.85	646	43.54
	20kroD100	342	2421	1.50	1	0.00	31	3.33	155	11.59
	20kroE100	154	1397	0.93	1	0.00	11	2.13	109	19.38
	20rat99	395	975	0.23	16	0.00	38	1.00	405	39.81
	20rd100	265	2561	1.62	0	0.00	9	1.86	128	5.44
	21eil101	356	3269	3.93	2	0.00	32	3.90	165	9.23
	21lin105	392	3677	2.01	4	0.00	9	2.48	177	2.47
	22pr107	253	2304	0.74	30	0.00	12	1.07	187	2.79
	25pr124	524	8959	9.81	4	0.00	13	15.41	131	11.97
	26bier127	530	9626	25.04	5	0.00	40	14.89	419	40.26
	26ch130	493	9396	24.93	1	0.00	23	14.11	282	79.01
	28pr136	491	4656	6.58	0	0.00	57	4.48	243	31.63
	29pr144	339	7470	15.52	0	0.00	9	9.13	319	7.80
	30ch150	657	10182	25.18	0	0.00	39	10.80	270	95.81
	30kroA150	436	10989	33.92	2	0.00	10	22.80	210	5.74
	30kroB150	395	6989	22.99	2	0.00	64	22.72	464	271.12
	31pr152	434	9557	18.94	1	0.00	9	10.56	251	8.01
	32u159	392	6500	14.98	1	0.00	11	6.06	143	7.72
	39rat195	655	9809	25.55	1	0.00	57	7.09	477	71.29
	40d198	744	6984	11.29	0	0.00	113	7.95	486	195.61
AVG		**570.67**	**5096.17**	**9.22**	**3.22**	**0.00**	**42.48**	**8.61**	**312.09**	**25.86**

Table 3: Branch-and-cut statistics on the instances with $\omega = 0.4$ and profit g_2.

		$\omega = 0.6$ and g_1								
		Nnodes	**Subtour**		**Conditional**		**Cover**		**Path**	
	Instance		**Num**	**Time**	**Num**	**Time**	**Num**	**Time**	**Num**	**Time**
	11berlin52	214	813	0.12	0	0.00	14	0.72	100	0.80
	11eil51	105	509	0.09	0	0.00	12	0.33	65	0.50
	14st70	226	1316	0.44	2	0.00	11	0.76	109	2.90
	16eil76	167	1466	0.64	3	0.00	6	0.79	54	0.47
	16pr76	321	2092	1.24	3	0.00	16	2.50	158	2.23
	20kroA100	490	5432	5.55	2	0.00	7	8.74	80	1.81
	20kroB100	428	4196	4.48	0	0.00	11	5.95	144	1.03
Set1	20kroC100	425	4048	4.02	1	0.00	24	5.86	261	2.23
	20kroD100	451	4369	3.64	1	0.00	9	5.67	105	3.26
	20kroE100	477	3959	2.43	0	0.00	4	4.29	46	0.49
	20rat99	313	2436	1.09	1	0.00	3	1.30	36	0.86
	20rd100	781	6982	7.07	0	0.00	23	6.55	176	5.94
	21eil101	450	4324	5.39	4	0.00	19	3.38	126	3.21
	21lin105	752	5506	3.84	7	0.00	9	6.23	90	1.18
	22pr107	700	7290	4.88	0	0.00	8	5.65	246	1.83
	11berlin52	7	39	0.01	1	0.00	0	0.03	0	0.00
	11eil51	82	544	0.13	0	0.00	4	0.36	34	2.42
	14st70	336	2471	1.16	6	0.00	8	1.32	64	1.15
	16eil76	250	2542	1.67	3	0.00	6	1.58	5	4.12
	16pr76	316	2895	1.97	4	0.00	9	3.74	87	1.61
	20kroA100	356	5927	7.75	2	0.00	21	10.43	276	9.97
	20kroB100	336	5446	6.36	0	0.00	5	8.54	133	1.36
Set2	20kroC100	328	4693	6.68	6	0.00	13	11.55	155	8.48
	20kroD100	420	5592	7.27	2	0.00	12	9.83	297	6.38
	20kroE100	314	3121	2.81	2	0.00	13	4.98	80	16.65
	20rat99	419	2929	2.01	2	0.00	3	1.99	21	53.31
	20rd100	313	4697	7.17	5	0.00	6	6.07	78	1.23
	21eil101	262	3696	5.72	0	0.00	3	2.71	60	2.99
	21lin105	515	7737	7.59	1	0.00	3	10.55	45	1.10
	22pr107	96	596	0.43	0	0.00	20	0.59	312	6.99
AVG		**355.00**	**3588.77**	**3.45**	**1.93**	**0.00**	**10.07**	**4.43**	**114.77**	**4.88**

		$\omega = 0.6$ and g_2								
		Nnodes	**Subtour**		**Conditional**		**Cover**		**Path**	
	Instance		**Num**	**Time**	**Num**	**Time**	**Num**	**Time**	**Num**	**Time**
	11berlin52	258	1063	0.17	5	0.00	10	1.00	67	0.37
	11eil51	73	414	0.08	0	0.00	4	0.21	53	0.41
	14st70	254	1075	0.34	1	0.00	25	0.82	292	6.31
	16eil76	224	1597	0.71	1	0.00	3	0.78	21	0.34
	16pr76	764	4298	2.59	5	0.00	54	5.80	210	6.45
	20kroA100	576	5376	5.60	0	0.00	37	9.47	273	7.17
	20kroB100	575	5233	5.57	1	0.00	16	8.84	130	1.67
Set1	20kroC100	886	7800	7.97	5	0.00	37	12.03	200	7.86
	20kroD100	597	4214	3.43	1	0.00	39	6.09	396	9.01
	20kroE100	457	3425	2.21	0	0.00	7	3.43	147	1.53
	20rat99	311	2443	1.09	2	0.00	5	1.41	70	0.72
	20rd100	764	4193	4.34	0	0.00	71	6.76	609	20.49
	21eil101	966	6079	7.53	0	0.00	84	7.31	304	22.71
	21lin105	837	5216	3.69	0	0.00	14	7.10	160	3.75
	22pr107	734	7588	5.06	8	0.00	17	6.19	464	4.05
	11berlin52	7	39	0.01	1	0.00	0	0.03	0	0.00
	11eil51	21	104	0.02	0	0.00	3	0.09	17	0.29
	14st70	420	3085	1.58	1	0.00	3	1.59	30	0.69
	16eil76	222	2109	1.34	4	0.00	8	1.25	47	1.24
	16pr76	260	2112	1.48	0	0.00	4	2.30	69	1.13
	20kroA100	335	5668	7.73	0	0.00	7	12.23	107	29.13
	20kroB100	453	6857	9.04	4	0.00	13	13.94	254	5.15
Set2	20kroC100	348	5350	7.04	0	0.00	12	8.48	134	5.07
	20kroD100	372	5380	6.29	2	0.00	22	8.76	131	7.37
	20kroE100	408	5533	5.84	13	0.00	11	7.05	124	3.12
	20rat99	493	3264	2.18	1	0.00	3	2.36	96	78.91
	20rd100	310	5185	8.18	2	0.00	8	6.50	104	1.32
	21eil101	453	7298	11.22	4	0.00	7	5.98	67	22.19
	21lin105	631	9487	11.18	13	0.00	10	10.12	302	6.89
	22pr107	460	5835	5.73	44	0.00	53	4.85	495	28.38
AVG		**448.97**	**4244.00**	**4.31**	**3.93**	**0.00**	**19.57**	**5.43**	**179.10**	**9.46**

Table 4: Branch-and-cut statistics on the instances with $\omega = 0.6$ and profit g_1 and g_2.

	$\omega = 0.8$ and g_1										
		Nnodes	Subtour		Conditional		Cover		Path		
	Instance		Num	Time	Num	Time	Num	Time	Num	Time
Set1	11berlin52	269	1286	0.22	22	0.00	10	1.03	81	0.45
	11eil51	101	472	0.10	1	0.00	5	0.23	31	0.15
	14st70	319	2313	0.97	2	0.00	4	1.29	40	0.37
	16eil76	294	2533	1.35	2	0.00	12	1.31	90	1.12
	16pr76	365	3602	2.33	2	0.00	12	3.83	116	5.05
	20kroA100	570	4960	5.36	0	0.00	54	7.68	293	17.85
	20kroB100	1125	12442	15.30	1	0.00	28	16.56	328	61.90
	20kroC100	482	4720	5.17	5	0.00	45	6.17	462	21.03
	20kroD100	473	5581	6.00	1	0.00	16	5.71	226	3.02
	20kroE100	788	5783	6.20	6	0.00	53	9.63	406	23.34
	20rat99	737	6604	5.94	3	0.00	27	4.97	155	11.53
	20rd100	418	4687	5.74	1	0.00	13	5.91	133	1.45
	21eil101	523	6211	8.46	3	0.00	7	4.37	48	2.18
	21lin105	748	7544	10.64	10	0.00	27	10.56	193	6.55
	22pr107	908	14048	16.36	9	0.00	30	13.24	612	13.21
Set2	11berlin52	0	4	0.00	0	0.00	0	0.01	0	0.00
	11eil51	69	363	0.07	8	0.00	2	0.20	9	0.07
	14st70	92	599	0.25	3	0.00	6	0.51	49	0.56
	16eil76	83	821	0.46	16	0.00	4	0.59	41	1.02
	16pr76	130	1214	0.67	20	0.00	9	1.56	95	1.46
	20kroA100	80	972	1.16	4	0.00	9	1.62	85	2.50
	20kroB100	420	6160	8.85	14	0.00	10	13.27	118	2.95
	20kroC100	417	6221	9.89	1	0.00	6	13.63	102	2.37
	20kroD100	669	10003	14.61	17	0.00	2	18.23	17	0.60
	20kroE100	456	5856	7.20	1	0.00	19	10.64	258	8.40
	20rat99	304	4146	3.49	1	0.00	2	3.14	16	1.45
	20rd100	300	4728	6.79	19	0.00	9	5.05	162	3.30
	21eil101	327	5445	8.45	6	0.00	3	4.32	33	10.76
	21lin105	36	329	0.39	1	0.00	4	0.52	48	0.87
	22pr107	45	408	0.52	3	0.00	9	0.60	95	2.91
AVG		**384.93**	**4335.17**	**5.10**	**6.07**	**0.00**	**14.57**	**5.55**	**144.73**	**6.95**

	$\omega = 0.8$ and g_2										
		Nnodes	Subtour		Conditional		Cover		Path		
	Instance		Num	Time	Num	Time	Num	Time	Num	Time
Set1	11berlin52	384	1568	0.28	27	0.00	7	1.22	51	0.55
	11eil51	228	1091	0.24	1	0.00	9	0.55	27	0.50
	14st70	434	2729	1.23	14	0.00	5	1.41	41	0.32
	16eil76	300	2120	1.17	0	0.00	10	1.18	86	1.51
	16pr76	588	5379	3.92	8	0.00	14	6.25	89	2.00
	20kroA100	920	7834	8.65	0	0.00	74	12.71	486	27.40
	20kroB100	842	11643	13.59	1	0.00	25	14.68	181	2.77
	20kroC100	811	6911	8.04	3	0.00	43	8.73	394	23.33
	20kroD100	559	8551	9.20	1	0.00	12	8.89	103	2.97
	20kroE100	720	5812	6.43	0	0.00	31	8.23	718	25.94
	20rat99	1111	9701	8.69	6	0.00	31	7.22	287	10.00
	20rd100	530	8827	10.87	1	0.00	8	12.51	113	8.27
	21eil101	1048	9168	13.65	0	0.00	38	9.02	113	32.38
	21lin105	814	11973	15.85	2	0.00	12	14.93	77	3.44
	22pr107	750	10736	12.13	24	0.00	16	10.07	290	5.17
Set2	11berlin52	2	6	0.00	0	0.00	0	0.01	0	0.00
	11eil51	34	235	0.05	1	0.00	1	0.14	6	0.03
	14st70	220	2137	0.84	0	0.00	4	1.29	35	0.26
	16eil76	350	3974	2.79	8	0.00	3	2.57	32	4.96
	16pr76	510	5235	3.55	26	0.00	8	6.48	76	2.93
	20kroA100	498	6557	8.70	7	0.00	44	13.67	222	19.35
	20kroB100	561	6972	8.69	0	0.00	12	12.10	146	3.27
	20kroC100	397	6195	7.81	2	0.00	11	11.45	151	3.79
	20kroD100	734	10636	13.71	29	0.00	17	20.35	185	4.13
	20kroE100	581	8520	10.23	0	0.00	7	16.56	110	1.61
	20rat99	286	3537	3.92	4	0.00	10	3.00	117	29.47
	20rd100	219	2566	3.48	4	0.00	11	3.38	111	2.73
	21eil101	620	9369	15.33	10	0.00	7	8.14	87	53.05
	21lin105	761	9707	14.16	21	0.00	27	17.60	281	10.86
	22pr107	316	2880	3.74	1	0.00	26	3.77	383	8.51
AVG		**537.60**	**6085.63**	**7.03**	**6.70**	**0.00**	**17.43**	**7.94**	**166.60**	**9.72**

Table 5: Branch-and-cut statistics on the instances with $\omega = 0.8$ and profit g_1 and g_2.

		g_1						g_2					
		clucut			BC			clucut			BC		
	Instance	Sol	Time	Gap%	Sol	Time	Gap%	Sol	Time	Gap%	Sol	Time	Gap%
							$\omega = 0.4$						
Set1	11berlin52	37	0.56	0.00%	37	0.41	0.00%	1829	0.64	0.00%	1829	0.46	0.00%
	11eil51	24	0.14	0.00%	24	0.21	0.00%	1279	0.26	0.00%	1279	1.03	0.00%
	14st70	33	0.45	0.00%	33	0.48	0.00%	1672	0.56	0.00%	1672	0.48	0.00%
	16eil76	40	3.38	0.00%	40	2.54	0.00%	2223	3.40	0.00%	2223	1.74	0.00%
	16pr76	47	4.57	0.00%	47	5.19	0.00%	2449	13.97	0.00%	2449	3.75	0.00%
	20kroA100	42	40.14	0.00%	42	27.14	0.00%	2151	69.19	0.00%	2151	30.50	0.00%
	20kroB100	49	13.84	0.00%	49	8.32	0.00%	2431	24.68	0.00%	2431	15.81	0.00%
	20kroC100	42	2.72	0.00%	42	2.07	0.00%	2174	2.62	0.00%	2174	4.64	0.00%
	20kroD100	39	3.23	0.00%	39	2.94	0.00%	1740	8.50	0.00%	1740	7.84	0.00%
	20kroE100	52	3.55	0.00%	52	3.68	0.00%	2415	2.49	0.00%	2415	5.18	0.00%
	20rat99	37	0.82	0.00%	37	1.61	0.00%	1905	0.75	0.00%	1905	0.59	0.00%
	20rd100	45	5.20	0.00%	45	5.62	0.00%	2228	13.91	0.00%	2228	11.38	0.00%
	21eil101	67	43.13	0.00%	67	12.20	0.00%	3365	56.74	0.00%	3365	15.78	0.00%
	21lin105	50	16.20	0.00%	50	31.96	0.00%	2489	12.82	0.00%	2489	12.89	0.00%
	22pr107	41	0.03	0.00%	41	0.04	0.00%	2123	0.05	0.00%	2123	0.10	0.00%
	25pr124	46	2130.68	0.00%	46	105.37	0.00%	2302	3516.36	0.00%	2302	175.17	0.00%
	26bier127	109	3759.00	8.49%	110	924.20	0.00%	5069	3685.02	15.49%	5420	2739.41	0.00%
	26ch130	70	3731.59	16.82%	70	499.02	0.00%	3423	3423.03	0.00%	3423	878.49	0.00%
	28pr136	53	281.85	0.00%	53	32.23	0.00%	2699	438.36	0.00%	2699	312.89	0.00%
	29pr144	6	3662.96	94.06%	60	1535.54	0.00%	3055	3767.94	39.15%	3055	1622.75	0.00%
	30ch150	61	3726.14	4.31%	61	570.16	0.00%	3131	1665.39	0.00%	3131	514.97	0.00%
	30kroA150	58	3742.55	28.62%	58	617.23	0.00%	3039	3732.85	12.73%	3039	742.87	0.00%
	30kroB150	66	3719.59	10.49%	66	343.15	0.00%	3172	3727.47	23.22%	3172	1933.77	0.00%
	31pr152	9	3651.60	91.43%	57	882.45	0.00%	2440	3649.65	54.71%	2915	1433.18	0.00%
	32u159	76	1710.73	0.00%	76	1296.21	0.00%	4002	2432.60	0.00%	4002	547.70	0.00%
	39rat195	71	1308.45	0.00%	71	287.79	0.00%	3656	975.52	0.00%	3656	251.21	0.00%
	40d198	70	501.47	0.00%	70	85.84	0.00%	3595	532.26	0.00%	3595	131.35	0.00%
Set2	11berlin52	50	0.85	0.00%	50	0.92	0.00%	2584	0.65	0.00%	2584	0.86	0.00%
	11eil51	37	0.32	0.00%	37	2.29	0.00%	1929	0.22	0.00%	1929	0.59	0.00%
	14st70	56	1.77	0.00%	56	0.76	0.00%	2736	1.67	0.00%	2736	0.84	0.00%
	16eil76	51	3.25	0.00%	51	2.70	0.00%	2518	5.58	0.00%	2518	10.63	0.00%
	16pr76	70	115.29	0.00%	70	129.47	0.00%	3550	107.33	0.00%	3550	30.61	0.00%
	20kroA100	80	1159.43	0.00%	80	40.97	0.00%	3894	681.68	0.00%	3894	54.43	0.00%
	20kroB100	86	533.20	0.00%	86	50.21	0.00%	4357	542.72	0.00%	4357	394.08	0.00%
	20kroC100	72	113.76	0.00%	72	27.22	0.00%	3586	175.16	0.00%	3586	95.66	0.00%
	20kroD100	78	24.67	0.00%	78	9.97	0.00%	3799	93.56	0.00%	3799	31.96	0.00%
	20kroE100	90	144.30	0.00%	90	7.54	0.00%	4614	21.82	0.00%	4614	27.59	0.00%
	20rat99	73	0.32	0.00%	73	1.69	0.00%	3624	1.06	0.00%	3624	42.50	0.00%
	20rd100	82	40.18	0.00%	82	27.52	0.00%	4181	39.30	0.00%	4181	28.32	0.00%
	21eil101	83	46.59	0.00%	83	29.75	0.00%	4264	71.45	0.00%	4264	45.93	0.00%
	21lin105	95	633.97	0.00%	95	329.39	0.00%	4814	748.22	0.00%	4814	339.21	0.00%
	22pr107	94	9.54	0.00%	94	13.10	0.00%	4740	69.03	0.00%	4740	18.69	0.00%
	25pr124	90	3624.48	25.62%	101	697.18	0.00%	4334	3622.27	28.41%	3859	3623.57	36.26%
	26bier127	11	3653.82	91.27%	124	3654.25	1.59%	6236	3673.75	1.53%	6004	3636.61	5.20%
	26ch130	6	3627.94	95.35%	111	2190.43	0.00%	250	3626.31	96.16%	5354	3635.65	16.25%
	28pr136	120	2239.39	0.00%	120	36.30	0.00%	6106	1609.05	0.00%	6106	146.88	0.00%
	29pr144	24	3636.55	83.22%	4	3629.55	97.20%	166	3629.58	97.71%	166	3626.60	97.71%
	30ch150	111	3627.12	25.50%	114	525.90	0.00%	124	3635.22	98.35%	6025	995.18	0.00%
	30kroA150	11	3624.33	92.62%	99	3631.89	33.56%	141	3628.84	98.13%	4478	3634.65	39.94%
	30kroB150	9	3629.74	93.92%	117	3640.74	19.16%	171	3627.88	97.73%	6190	3625.89	17.73%
	31pr152	89	3631.77	40.67%	9	3629.77	94.00%	431	3637.55	94.34%	431	3631.17	94.34%
	32u159	143	3627.89	7.14%	143	428.88	0.00%	7507	3620.71	4.37%	7507	919.53	0.00%
	39rat195	135	750.97	0.00%	135	472.05	0.00%	6813	1201.13	0.00%	6813	292.75	0.00%
	40d198	149	3432.08	0.00%	149	2728.54	0.00%	6700	3630.10	26.78%	7480	303.29	0.00%
AVG			**1370.33**	**14.99%**		**615.23**	**4.55%**		**1360.35**	**14.61%**		**751.66**	**5.69%**
#Opt				**38**			**49**			**39**			**47**

Table 6: Comparison between **clucut** and BC on the *Set1* and *Set2* instances with $\omega = 0.4$.

$\omega = 0.6$													
		g_1						g_2					
		clucut			BC			clucut			BC		
	Instance	**Sol**	**Time**	**Gap%**	**Sol**	**Time**	**Gap%**	**Sol**	**Time**	**Gap%**	**Sol**	**Time**	**Gap%**
Set1	11berlin52	43	3.41	0.00%	43	3.88	0.00%	2190	2.12	0.00%	2190	3.72	0.00%
	11eil51	39	0.53	0.00%	39	1.81	0.00%	1911	0.60	0.00%	1911	1.21	0.00%
	14st70	50	80.90	0.00%	50	19.87	0.00%	2589	33.47	0.00%	2589	18.62	0.00%
	16eil76	59	59.75	0.00%	59	8.02	0.00%	3119	63.80	0.00%	3119	19.60	0.00%
	16pr76	65	59.88	0.00%	65	124.85	0.00%	3275	1089.40	0.00%	3275	182.03	0.00%
	20kroA100	65	1573.50	0.00%	65	106.66	0.00%	3192	1524.53	0.00%	3192	135.17	0.00%
	20kroB100	59	3631.79	36.39%	66	97.08	0.00%	3203	1723.12	0.00%	3203	165.41	0.00%
	20kroC100	62	424.37	0.00%	62	72.43	0.00%	3110	1321.52	0.00%	3110	246.28	0.00%
	20kroD100	64	1901.65	0.00%	64	75.22	0.00%	3133	1740.32	0.00%	3133	81.68	0.00%
	20kroE100	63	84.31	0.00%	63	166.56	0.00%	2950	237.76	0.00%	2950	82.57	0.00%
	20rat99	52	102.58	0.00%	52	46.19	0.00%	2643	66.64	0.00%	2643	41.06	0.00%
	20rd100	72	321.63	0.00%	72	368.21	0.00%	3591	265.07	0.00%	3591	133.94	0.00%
	21eil101	82	762.19	0.00%	82	82.72	0.00%	4187	602.27	0.00%	4187	412.53	0.00%
	21lin105	78	472.41	0.00%	78	132.16	0.00%	3955	1087.02	0.00%	3955	162.10	0.00%
	22pr107	53	3622.72	36.14%	53	3625.48	31.17%	2697	3626.63	34.73%	2697	3626.40	30.44%
Set2	11berlin52	51	0.11	0.00%	51	0.14	0.00%	2608	0.11	0.00%	2608	0.14	0.00%
	11eil51	50	0.57	0.00%	50	3.76	0.00%	2575	0.51	0.00%	2575	0.56	0.00%
	14st70	64	1360.19	0.00%	64	300.55	0.00%	3218	2602.34	0.00%	3218	502.19	0.00%
	16eil76	74	353.70	0.00%	74	175.62	0.00%	3728	89.88	0.00%	3728	100.66	0.00%
	16pr76	74	3620.91	1.33%	74	1777.29	0.00%	3729	3334.55	0.00%	3729	481.00	0.00%
	20kroA100	91	3625.62	8.08%	95	3623.23	4.04%	3763	3628.82	24.86%	4554	3621.34	9.07%
	20kroB100	93	3630.43	6.06%	2	3621.31	97.98%	3578	3630.00	28.55%	4668	3623.16	6.79%
	20kroC100	5	3626.30	94.95%	90	3618.66	9.09%	3915	3622.41	21.83%	4534	3619.27	9.46%
	20kroD100	4	3624.16	95.96%	93	3618.94	6.06%	4394	3628.09	12.26%	4570	3618.92	8.75%
	20kroE100	97	3620.09	2.02%	97	2215.67	0.00%	4910	3621.33	1.96%	4910	3617.95	1.96%
	20rat99	87	140.58	0.00%	87	194.56	0.00%	4516	70.56	0.00%	4516	154.69	0.00%
	20rd100	97	3626.93	2.02%	99	2135.44	0.00%	5008	2876.95	0.00%	4957	3619.27	1.02%
	21eil101	95	3622.50	5.00%	97	962.97	0.00%	4933	3621.81	2.32%	4933	3622.82	2.32%
	21lin105	102	3641.03	1.92%	104	781.43	0.00%	5075	3629.93	2.93%	5103	3627.18	2.39%
	22pr107	106	225.08	0.00%	106	11.02	0.00%	5363	28.98	0.00%	5363	134.78	0.00%
AVG			**1593.99**	**9.66%**		**932.39**	**4.94%**		**1592.35**	**4.31%**		**1188.54**	**2.41%**
#Opt				**19**			**25**			**22**			**21**

Table 7: Comparison between **clucut** and BC on the *Set1* and *Set2* instances with $\omega = 0.6$.

time for BC is equal to 615 and 751 seconds for the instances with profits g_1 and g_2, respectively. On the contrary, for **clucut** this time is equal to 1370 and 1360 seconds, respectively. Regarding the effectiveness, the average Gap% values are equal to 4.55% and 5.69% for BC and equal to 14.99% and 14.61% for **clucut**. Moreover, from line #Opt we observe that BC is able to solve 49 instances to optimality (out of 54) for g_1 and 47 for g_2, while **clucut** solves 38 instances for g_1 and 39 for g_2. It is worth noting that BC optimally solves all the instances of *Set1* while it does not find the optimal solution 12 times on the *Set2* dataset. This shows that, for our algorithm, the instances of *Set2* are more difficult to solve than the ones of *Set1*. We recall that the only difference is that customers are randomly assigned to clusters in *Set2*, while they are geographically clustered in *Set1*.

Similar considerations can be done for the results reported in Table 7

$\omega = 0.8$														
		g_1						g_2						
		clucut			BC			clucut			BC			
	Instance	Sol	Time	Gap%	Sol	Time	Gap%	Sol	Time	Gap%	Sol	Time	Gap%
Set1	11berlin52	47	54.39	0.00%	47	6.36	0.00%	2384	11.80	0.00%	2384	12.23	0.00%
	11eil51	43	4.06	0.00%	43	2.10	0.00%	2114	6.50	0.00%	2114	6.94	0.00%
	14st70	65	773.70	0.00%	65	361.26	0.00%	3355	488.31	0.00%	3355	533.94	0.00%
	16eil76	69	476.84	0.00%	69	153.38	0.00%	3573	1334.32	0.00%	3573	85.04	0.00%
	16pr76	72	3618.59	1.37%	72	1653.24	0.00%	3611	3625.51	2.58%	3611	3619.83	2.25%
	20kroA100	73	3630.08	26.26%	79	224.73	0.00%	2713	3631.67	45.83%	4115	3034.79	0.00%
	20kroB100	77	3636.04	22.22%	86	2802.72	0.00%	4188	3627.16	16.37%	4117	3637.40	16.25%
	20kroC100	76	3630.97	23.23%	83	417.48	0.00%	3999	3624.35	20.15%	3999	287.95	0.00%
	20kroD100	77	3633.75	22.22%	85	430.87	0.00%	3854	3630.75	23.04%	4026	3625.22	19.61%
	20kroE100	78	3627.27	18.75%	80	347.91	0.00%	4002	2849.01	0.00%	4002	388.90	0.00%
	20rat99	69	3632.64	21.59%	79	1810.70	0.00%	3855	3625.82	13.06%	3992	2785.76	0.00%
	20rd100	90	3627.83	9.09%	91	813.84	0.00%	3892	3634.55	22.28%	4640	3626.99	7.35%
	21eil101	89	3630.17	11.00%	91	319.53	0.00%	4538	3635.25	10.14%	4717	1786.74	0.00%
	21lin105	87	3639.87	16.35%	90	289.91	0.00%	4245	3648.18	18.80%	4561	3638.53	10.43%
	22pr107	6	3637.27	94.34%	53	3645.25	50.00%	2156	3639.16	59.80%	2697	3636.09	49.71%
Set2	11berlin52	51	0.02	0.00%	51	0.03	0.00%	2608	0.06	0.00%	2608	0.07	0.00%
	11eil51	50	1.13	0.00%	50	0.79	0.00%	2575	0.45	0.00%	2575	0.51	0.00%
	14st70	69	7.23	0.00%	69	3.57	0.00%	3513	22.52	0.00%	3513	13.25	0.00%
	16eil76	75	12.27	0.00%	75	4.03	0.00%	3800	3.31	0.00%	3800	167.70	0.00%
	16pr76	75	1997.86	0.00%	75	8.15	0.00%	3800	2003.37	0.00%	3800	670.13	0.00%
	20kroA100	99	331.47	0.00%	99	9.73	0.00%	4086	3627.77	18.41%	4241	3623.81	15.32%
	20kroB100	69	3634.85	30.30%	99	1207.78	0.00%	83	3633.52	98.34%	4668	3624.05	6.79%
	20kroC100	4	3631.41	95.96%	94	3623.28	5.05%	249	3641.88	95.03%	3043	3622.69	39.24%
	20kroD100	5	3630.36	94.95%	95	3631.99	4.04%	3750	3624.62	25.12%	4776	3634.34	4.63%
	20kroE100	97	3624.73	2.02%	98	3620.10	1.01%	325	3631.13	93.51%	325	3627.43	93.51%
	20rat99	98	1335.88	0.00%	98	155.44	0.00%	5007	531.44	0.00%	5007	323.20	0.00%
	20rd100	99	134.14	0.00%	99	135.97	0.00%	5008	24.29	0.00%	5008	45.00	0.00%
	21eil101	99	3623.63	1.00%	100	1569.38	0.00%	4831	3628.04	4.34%	4933	3629.89	2.32%
	21lin105	104	3.76	0.00%	104	4.51	0.00%	5228	1953.40	0.00%	5228	1541.45	0.00%
	22pr107	106	11.65	0.00%	106	8.74	0.00%	5363	62.54	0.00%	5363	138.40	0.00%
AVG			**2107.80**	**16.36%**		**908.76**	**2.00%**		**2246.69**	**18.89%**		**1845.61**	**8.91%**
#Opt				**14**			**26**			**14**			**18**

Table 8: Comparison between **clucut** and BC on the *Set1* and *Set2* instances with $\omega = 0.8$.

which are related to $\omega = 0.6$. Here we report the results for instances with up to 107 customers as no meaningful result was obtained for larger sizes. BC solves 25 instances to optimality (out of 30) for g_1 and 21 for g_2, while **clucut** solves 19 and 22, respectively. Also, as before, BC performs much better in terms of both computational time and optimality gap. More in detail, the Gap% value for BC is equal to 4.94% and 2.41% for g_1 and g_2, respectively, while it is around double for **clucut**. Regarding the computational time, BC is nearly 41% faster than **clucut** for g_1 and 25% faster for g_2. It is interesting to note that instances with $\omega = 0.6$ are more difficult to solve than those with $\omega = 0.4$. Indeed, for both algorithms, the number of instances optimally solved decreases while the computational times increase. This was expected: the larger is the value of ω, the larger is the set of feasible solutions so the more difficult is the problem to solve.

The results of Table 8, which are related to $\omega = 0.8$, show that the performance of **clucut** largely deteriorates with respect to the case $\omega = 0.6$. In fact, the number of instances solved to optimality reduces to 14 for both g_1 and g_2 while the average optimality gap increases to 16.36% for g_1 and 18.89% for g_2. There is also an increase of the computational time of **clucut**, which now exceeds 2100 seconds. On the contrary, the results of BC are more stable. Indeed it solves 26 instances to optimality for g_1 and 18 for g_2. Once again BC largely outperforms **clucut** in both computational time and optimality gap at termination. It is worth noting that, when going from $\omega = 0.6$ to $\omega = 0.8$, the average time of BC with profit g_1 does not significantly change and the optimality gap reduces from 4.94% to 2.00%. On the contrary, for profit g_2, both the computational time and the optimality gap remarkably increase. Therefore, when going from $\omega = 0.6$ to $\omega = 0.8$, there is a significant difference in the behaviour of the algorithm with respect to the type of profit considered.

Thus, by summarizing all results in Tables 6–8, except for cluster cover inequalities, we can conclude that the valid inequalities described in Section 3 greatly improves the efficacy of the exact algorithm.

5.3 Comparison with Benchmark Approaches

In this section, we compare BC with two exact approaches proposed in the literature, specifically, the one proposed in [2] (called **ACC** from now on) and the one proposed in [20] (called **PFS**). It is worth noting that BC and **ACC** were executed on the same machine and then their CPU times are directly comparable. For **PFS**, we scale the CPU time according to the processor performance.

In Tables 9 and 10 we compare the solutions found by BC with the **ACC** algorithm proposed in [2] on the dataset *Set1* and *Set2*, respectively. The formulation used in **ACC** is a polynomial formulation where subtour elimination constraints are modelled as Miller-Tucker-Zemlin (MTZ) constraints. The formulation uses three types of binary variables: arc variables, vertex visiting variables and cluster visiting variables. The results are restricted to the instances with 100 customers at most as **ACC** was not capable of providing any solution within the time limit for larger instances. The first three columns of the Tables report the name of the instance (*Instance*), the ω value (ω) and the type of profit (p_g). The next six columns are grouped in two parts referring to the results by **ACC**, first, and BC, second. Specifically, we report the value of the best solution found at termination (*Sol*), the

	Instance	ω	p_g	ACC Sol	ACC Time	ACC Gap%	BC Sol	BC Time	BC Gap%
Set1	11berlin52	0.4	g_1	37	15.52	0.00%	37	0.41	0.00%
	11berlin52	0.4	g_2	1829	18.14	0.00%	1829	0.46	0.00%
	11berlin52	0.6	g_1	43	646.07	0.00%	43	3.88	0.00%
	11berlin52	0.6	g_2	2190	540.54	0.00%	2190	3.72	0.00%
	11berlin52	0.8	g_1	47	1877.97	0.00%	47	6.36	0.00%
	11berlin52	0.8	g_2	2384	1207.51	0.00%	2384	12.23	0.00%
	11eil51	0.4	g_1	24	12.51	0.00%	24	0.21	0.00%
	11eil51	0.4	g_2	1279	15.90	0.00%	1279	1.03	0.00%
	11eil51	0.6	g_1	39	9.38	0.00%	39	1.81	0.00%
	11eil51	0.6	g_2	1911	75.43	0.00%	1911	1.21	0.00%
	11eil51	0.8	g_1	43	1987.21	0.00%	43	2.10	0.00%
	11eil51	0.8	g_2	2114	1673.87	0.00%	2114	6.94	0.00%
	14st70	0.4	g_1	33	3610.11	18.98%	33	0.48	0.00%
	14st70	0.4	g_2	1672	3610.11	14.62%	1672	0.48	0.00%
	14st70	0.6	g_1	50	3610.11	14.35%	50	19.87	0.00%
	14st70	0.6	g_2	2589	3610.11	14.18%	2589	18.62	0.00%
	14st70	0.8	g_1	64	3610.10	5.88%	65	361.26	0.00%
	14st70	0.8	g_2	3229	3610.10	7.43%	3355	533.94	0.00%
	16eil76	0.4	g_1	40	2161.31	0.00%	40	2.54	0.00%
	16eil76	0.4	g_2	2223	2142.87	0.00%	2223	1.74	0.00%
	16eil76	0.6	g_1	59	3610.10	9.41%	59	8.02	0.00%
	16eil76	0.6	g_2	3119	3610.10	8.63%	3119	19.60	0.00%
	16eil76	0.8	g_1	67	3610.10	8.22%	69	153.38	0.00%
	16eil76	0.8	g_2	3525	3610.11	6.60%	3573	85.04	0.00%
	16pr76	0.4	g_1	47	3610.10	20.18%	47	5.19	0.00%
	16pr76	0.4	g_2	2449	3609.80	18.89%	2449	3.75	0.00%
	16pr76	0.6	g_1	65	3609.79	7.48%	65	124.85	0.00%
	16pr76	0.6	g_2	3275	3608.23	7.36%	3275	182.03	0.00%
	16pr76	0.8	g_1	71	3609.71	4.05%	72	1653.24	0.00%
	16pr76	0.8	g_2	3601	3608.42	3.43%	3611	3619.83	2.25%
	20kroA100	0.4	g_1	42	3609.53	26.60%	42	27.14	0.00%
	20kroA100	0.4	g_2	2151	3609.85	28.60%	2151	30.50	0.00%
	20kroA100	0.6	g_1	65	3609.89	18.68%	65	106.66	0.00%
	20kroA100	0.6	g_2	3164	3609.87	23.60%	3192	135.17	0.00%
	20kroA100	0.8	g_1	76	3610.03	23.23%	79	224.73	0.00%
	20kroA100	0.8	g_2	3919	3609.77	21.75%	4115	3034.79	0.00%
	AVG				**2550.01**	**8.67%**		**288.70**	**0.06%**
	#Opt					**14**			**35**

Table 9: Comparison between **ACC** and BC on the *Set1* instances.

computational time (*Time*), in seconds, and the optimality gap (*Gap%*) at termination (which is 0% in case the instance is solved to proven optimality). The last two rows of each table report the average computational time and the average optimality gap, and the number of instances solved to optimality, respectively.

The results of Table 9 highlight that BC is much better than **ACC** from the effectiveness and performance point of view. The AVG line shows that the average gap of BC is equal to 0.06% and 35 out of 36 instances are optimally solved. In the only case in which BC fails to find the optimal solution, the Gap% value is equal to 2.25%. Instead, the average gap of **ACC** goes up to 8.67% and the algorithm optimally solves only 14 instances within the time limit of one hour. It is worth noting that in 12 out of 22 instances not optimally solved by **ACC**, the Gap% value is greater than 14% with a peak equal to 28.60%. Regarding the performance, with an average computational time equal to 288 seconds, BC is eight times faster than **ACC**. BC proves to be better than **ACC** even on dataset *Set2* but the gap is smaller here. Indeed, BC optimally solves 33 out of 36 instances while **ACC** solves 22 of them. The average gap is equal to 0.79% for BC and 3.49% for **ACC** and, about the performance, the average time of BC is equal to 426 seconds while it is 1644 seconds for **ACC**. Finally, in the worst case, the Gap% value is equal to 15.32% for BC and 24.50% for **ACC**.

The last comparison is carried out between BC and **PFS**. The formulation used in **PFS** involves only two types of binary variables, namely arc variables and vertex visiting variables. Subtour elimination constraints are dynamically separated. Moreover, a greedy construction procedure is used for creating an initial feasible solution which is used as a warm start for CPLEX. The results of the comparison are shown in Table 11 and are restricted to the instances tested in [20], i.e., 11berlin52, 11eil51, 14st70, 16eil76, for values of ω equal to 0.4, 0.6 and 0.8. The headings of Table 11 are the same as in Table 10, the only difference being that the Gap% column is removed because all the instances are optimally solved by both algorithms. Indeed, in [20] only instances solved to optimality are reported. In order to have a fair comparative from the performance point of view, the CPU time reported in [20] has been scaled according to the scaling factor reported here: `https://www.cpubenchmark.net/CPU_mega_page.html`. From the results of Table 11, we observe that BC is around 17% faster than **PFS**. A detailed analysis reveals that in 14 out of 20 instances, BC is faster than **PFS**. Essentially, **PFS** wins on the smallest instances where probably the branch-and-cut pays on overhead associated with the separation of the valid inequalities. This is certified by the results of the two algorithms on the last 11 instances of the table, the largest ones, where BC is faster than **PFS** in 10 cases.

	Instance	ω	p_g	ACC Sol	ACC Time	ACC Gap%	BC Sol	BC Time	BC Gap%
Set2	11berlin52	0.4	g_1	50	30.90	0.00%	50	0.92	0.00%
	11berlin52	0.4	g_2	2584	53.99	0.00%	2584	0.86	0.00%
	11berlin52	0.6	g_1	51	1.01	0.00%	51	0.14	0.00%
	11berlin52	0.6	g_2	2608	1.15	0.00%	2608	0.14	0.00%
	11berlin52	0.8	g_1	51	0.51	0.00%	51	0.03	0.00%
	11berlin52	0.8	g_2	2608	0.66	0.00%	2608	0.07	0.00%
	11eil51	0.4	g_1	37	3.98	0.00%	37	2.29	0.00%
	11eil51	0.4	g_2	1929	10.25	0.00%	1929	0.59	0.00%
	11eil51	0.6	g_1	50	17.71	0.00%	50	3.76	0.00%
	11eil51	0.6	g_2	2575	10.25	0.00%	2575	0.56	0.00%
	11eil51	0.8	g_1	50	4.72	0.00%	50	0.79	0.00%
	11eil51	0.8	g_2	2575	1.55	0.00%	2575	0.51	0.00%
	14st70	0.4	g_1	56	3609.95	12.50%	56	0.76	0.00%
	14st70	0.4	g_2	2736	3609.76	16.11%	2736	0.84	0.00%
	14st70	0.6	g_1	64	3609.40	7.25%	64	300.55	0.00%
	14st70	0.6	g_2	3218	3609.79	8.40%	3218	502.19	0.00%
	14st70	0.8	g_1	69	77.66	0.00%	69	3.57	0.00%
	14st70	0.8	g_2	3513	130.22	0.00%	3513	13.25	0.00%
	16eil76	0.4	g_1	51	1515.90	0.00%	51	2.70	0.00%
	16eil76	0.4	g_2	2518	3610.01	6.66%	2518	10.63	0.00%
	16eil76	0.6	g_1	74	1767.92	0.00%	74	175.62	0.00%
	16eil76	0.6	g_2	3728	1183.11	0.00%	3728	100.66	0.00%
	16eil76	0.8	g_1	75	249.92	0.00%	75	4.03	0.00%
	16eil76	0.8	g_2	3800	65.56	0.00%	3800	167.70	0.00%
	16pr76	0.4	g_1	70	3610.10	3.65%	70	129.47	0.00%
	16pr76	0.4	g_2	3402	3610.09	8.52%	3550	30.61	0.00%
	16pr76	0.6	g_1	74	3434.82	0.00%	74	1777.29	0.00%
	16pr76	0.6	g_2	3729	3610.10	1.66%	3729	481.00	0.00%
	16pr76	0.8	g_1	75	69.28	0.00%	75	8.15	0.00%
	16pr76	0.8	g_2	3800	21.90	0.00%	3800	670.13	0.00%
	20kroA100	0.4	g_1	75	3610.12	24.24%	80	40.97	0.00%
	20kroA100	0.4	g_2	3781	3610.11	24.50%	3894	54.43	0.00%
	20kroA100	0.6	g_1	95	3610.12	4.04%	95	3623.23	4.04%
	20kroA100	0.6	g_2	4741	3610.12	5.33%	4554	3621.34	9.07%
	20kroA100	0.8	g_1	98	3610.12	1.01%	99	9.73	0.00%
	20kroA100	0.8	g_2	4920	3610.12	1.76%	4241	3623.81	15.32%
	AVG				**1644.25**	**3.49%**		**426.76**	**0.79%**
	#Opt					**22**			**33**

Table 10: Comparison between **ACC** and BC on the *Set2* instances.

6 Conclusions

This paper deals with the Set Orienteering Problem (SOP), which is a variant of the Orienteering Problem recently introduced in the literature. Specifically, in the SOP, customers are grouped in clusters and the profit associated with a cluster is collected in case at least one customer from the cluster is visited. The SOP finds interesting applications in practice, especially related

	Instance	ω	p_g	PFS Sol	PFS Time	BC Sol	BC Time
Set1	11berlin52	0.4	g_1	37	0.78	37	0.41
	11berlin52	0.4	g_2	1829	0.85	1829	0.46
	11berlin52	0.6	g_1	43	3.05	43	3.88
	11berlin52	0.6	g_2	2190	0.96	2190	3.72
	11berlin52	0.8	g_1	47	3.33	47	6.36
	11berlin52	0.8	g_2	2384	5.52	2384	12.23
	11eil51	0.4	g_1	24	1.83	24	0.21
	11eil51	0.4	g_2	1279	2.02	1279	1.03
	11eil51	0.6	g_1	39	1.20	39	1.81
	11eil51	0.6	g_2	1911	2.17	1911	1.21
	11eil51	0.8	g_1	43	11.88	43	2.10
	11eil51	0.8	g_2	2114	29.01	2114	6.94
	14st70	0.4	g_1	33	11.98	33	0.48
	14st70	0.4	g_2	1672	20.50	1672	0.48
	14st70	0.8	g_1	65	690.35	65	361.26
	14st70	0.8	g_2	3355	164.63	3355	533.94
	16eil76	0.4	g_1	40	62.00	40	2.54
	16eil76	0.4	g_2	2223	27.01	2223	1.74
	16eil76	0.6	g_1	59	46.27	59	8.02
	16eil76	0.6	g_2	3119	78.24	3119	19.60
	AVG				**58.18**		**48.42**

Table 11: Comparison between **PFS** and BC

to mass distribution products. In general, routing problems with profits are facing a new wave of interest from the research community thanks to their link with fast delivery services, where it often happens that not all customers requiring a service can be satisfied, thus a selection is needed (which is the main feature of routing problems with profits). Specifically, the SOP finds applications in delivery services where multiple options are associated with each customer regarding where a parcel can be delivered.

We focus on developing an exact algorithm for the SOP. Specifically, we propose a new formulation for the problem that has fewer variables than those proposed in the literature as it does not include vertex visiting variables. We show that the new formulation has a stronger relaxation than the formulation with vertex visiting variables. Then, we propose different classes of valid inequalities to strengthen the formulation. Exhaustive computational tests show that the resulting branch-and-cut algorithm is effective. The performance depends on the number of customers and on the value of ω, which measures the maximum route length. Specifically, when ω is small (and the vehicle route is short), the branch-and-cut algorithm is able to solve almost all instances with up to 200 vertices. For larger values of ω, the algorithm fails to solve instances with more than 107 customers, while it solves the majority of those with fewer customers. Also, a comparison with two exact approaches proposed in the literature shows that our branch-and-cut algorithm scales better in terms of number of customers and can thus be considered as the new state-of-the art exact solution approach for the SOP.

As a future research direction, we plan to study the case in which multiple vehicles are available and to adapt the branch-and-cut algorithm to this case. Also, a column generation approach might be suitable in the multiple vehicle case. Finally, it might be interesting to investigate whether similar formulations as the one proposed in this paper (which get rid of the vertex visiting variables) are effective in the solution of similar problems, as, for example, the Cluster Orienteering Problem.

References

[1] E. Angelelli, C. Archetti, and M. Vindigni. The clustered orienteering problem. *European Journal of Operational Research*, 238:404–414, 2014.

[2] C. Archetti, F. Carrabs, and R. Cerulli. The set orienteering problem. *European Journal of Operational Research*, 267(1):264–272, 2018.

[3] C. Archetti, M.G. Speranza, and D. Vigo. Vehicle routing problems with profits. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications, Second Edition, MOS-SIAM Series on Optimization 18*, pages 273–298. SIAM, 2014.

[4] G. Best and G.A. Hollinger. Decentralised self-organising maps for the online orienteering problem with neighbourhoods. In *International Symposium on Multi-Robot and Multi-Agent Systems*, pages 139–141, 2019.

[5] F. Carrabs. A biased random-key genetic algorithm for the set orienteering problem. *European Journal of Operational Research*, 292(3):830–854, 2021.

[6] F. Carrabs, C. Cerrone, R. Cerulli, and C. D'Ambrosio. Improved upper and lower bounds for the close enough traveling salesman problem. *Lecture Notes in Computer Science (including subseries Lecture Notes*

in Artificial Intelligence and Lecture Notes in Bioinformatics), 10232 LNCS:165–177, 2017.

[7] F. Carrabs, C. Cerrone, R. Cerulli, and M. Gaudioso. A novel discretization scheme for the close enough traveling salesman problem. *Computers and Operations Research*, 78:163–171, 2017.

[8] F. Carrabs, C. Cerrone, R. Cerulli, and B.L. Golden. An adaptive heuristic approach to compute upper and lower bounds for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 32(4):1030–1048, 2020.

[9] M. Desrochers and G. Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

[10] B. Dezső, A. Jüttner, and P. Kovács. LEMON – an Open Source C++ Graph Template Library. *Electronic Notes in Theoretical Computer Science*, 264:23–45, 2011.

[11] A.G. Dragomir, T. Van Woensel, and K.F. Doerner. The pickup and delivery problem with alternative locations and overlapping time windows. *Computers & Operations Research*, 143:105758, 2022.

[12] D. Dumez, F. Lehuédé, and O. Péton. A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological*, 144:103–132, 2021.

[13] M. Fischetti, J.J. Salazar, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[14] M. Fischetti, J.J. Salazar, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1998.

[15] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32:263–273, 1998.

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

[17] Y. Li, C. Archetti, and I. Ljubic. Reinforcement learning approaches for the orienteering problem with stochastic and dynamic release dates. *arXiv preprint arXiv:2207.00885*, 2022.

[18] S. Mancini and M. Gansterer. Vehicle routing with private and shared delivery locations. *Computers & Operations Research*, 133:105361, 2021.

[19] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[20] R. Pěnička, J. Faigl, and M. Saska. Variable neighborhood search for the set orienteering problem and its application to other orienteering problem variants. *European Journal of Operational Research*, 276(3):816–825, 2019.

[21] R. Pěnička, J. Faigl, P. Váňa, and M. Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, 2017.

[22] R. Pěnička, J. Faigl, P. Váňa, and M. Saska. Dubins orienteering problem with neighborhoods. In *Proceedings of the International conference on unmanned aircraft systems*, pages 1555–1562, 2017.

[23] C. Tilk, K. Olkis, and S. Irnich. The last-mile vehicle routing problem with delivery options. *OR Spectrum*, 43(4):877–904, 2021.

[24] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809, 1984.

[25] M.W. Ulmer and S. Streng. Same-day delivery with pickup stations and autonomous vehicles. *Computers & Operations Research*, 108:1–19, 2019.

[26] S.A. Voccia, A.M. Campbell, and B.W. Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

[27] L. Zhou, R. Baldacci, D. Vigo, and X. Wang. A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, 265(2):765–778, 2018.

Appendix A Detailed Results

In this section we show the detailed results of the comparison among the branch-and-cut algorithm with the full set of inequalities and the versions in which we discard a single inequality at a time. These results are presented in Tables 12–15. The first column (*Instance*) shows the name of the instance. Then, five groups of columns follow, corresponding to the five versions of the branch-and-cut algorithm: the version with the full set of inequalities (*C-BC*), no conditional cuts (*NoCond*), no cover inequalities (*NoCover*), no cluster cover inequalities (*NoCluCover*) and no path inequalities (*NoPath*). For each version of the algorithm, we report the value of the upper bound (*UB*) at termination (which corresponds to the value of the optimal solution in case the computational time is lower than one hour) and the computational time (*Time*). Also, for each version of the algorithm in which one inequality is excluded, we report the percentage gap of the corresponding upper bound with respect to the upper bound obtained by C-BC, calculated as $gap = \frac{UB_{C-BC} - UB_*}{UB_{C-BC}}$, where UB_{C-BC} and UB_* are the upper bound of C-BC and of the version considered, respectively. Note that positive values of the gap mean that the version of the algorithm without the inequality gives a better result that C-BC. The second last row of each table reports the average values of time and gap while the last row report, for each version of the algorithm in which an inequality is excluded, the number of times in which the upper bound provided by that version is worse than the one provided by C-BC. These two rows are the same as the ones reported in Table 1.

$\omega = 0.4$ and g_1															
		C-BC		**NoCond**			**NoCover**			**NoCluCover**			**NoPath**		
	Instance	**UB**	**Time**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**
	11berlin52	37.0	0.39	37.0	0.52	0.00%	37.0	0.65	0.00%	37.0	0.41	0.00%	37.0	1.19	0.00%
	11eil51	24.0	0.34	24.0	0.34	0.00%	24.0	0.29	0.00%	24.0	0.21	0.00%	24.0	0.40	0.00%
	14st70	33.0	0.48	33.0	0.49	0.00%	33.0	0.58	0.00%	33.0	0.48	0.00%	33.0	0.70	0.00%
	16eil76	40.0	2.54	40.0	2.59	0.00%	40.0	1.52	0.00%	40.0	2.54	0.00%	40.0	4.24	0.00%
	16pr76	47.0	5.08	47.0	5.21	0.00%	47.0	8.04	0.00%	47.0	5.19	0.00%	47.0	7.46	0.00%
	20kroA100	42.0	27.09	42.0	27.97	0.00%	42.0	33.37	0.00%	42.0	27.14	0.00%	42.0	38.52	0.00%
	20kroB100	49.0	11.27	49.0	10.04	0.00%	49.0	15.13	0.00%	49.0	8.32	0.00%	49.0	22.92	0.00%
	20kroC100	42.0	2.05	42.0	2.04	0.00%	42.0	3.11	0.00%	42.0	2.07	0.00%	42.0	4.62	0.00%
	20kroD100	39.0	3.04	39.0	2.85	0.00%	39.0	4.26	0.00%	39.0	2.94	0.00%	39.0	7.11	0.00%
	20kroE100	52.0	5.07	52.0	4.39	0.00%	52.0	3.93	0.00%	52.0	3.68	0.00%	52.0	5.67	0.00%
	20rat99	37.0	1.60	37.0	0.78	0.00%	37.0	1.04	0.00%	37.0	1.61	0.00%	37.0	1.13	0.00%
	20rd100	45.0	5.99	45.0	5.76	0.00%	45.0	7.86	0.00%	45.0	5.62	0.00%	45.0	8.87	0.00%
	21eil101	67.0	23.12	67.0	30.11	0.00%	67.0	22.93	0.00%	67.0	12.20	0.00%	67.0	48.34	0.00%
Set1	21lin105	50.0	33.20	50.0	6.06	0.00%	50.0	38.34	0.00%	50.0	31.96	0.00%	50.0	15.34	0.00%
	22pr107	41.0	0.04	41.0	0.04	0.00%	41.0	0.03	0.00%	41.0	0.04	0.00%	41.0	0.04	0.00%
	25pr124	46.0	136.08	46.0	114.73	0.00%	46.0	2773.53	0.00%	46.0	105.37	0.00%	46.0	2443.04	0.00%
	26bier127	110.0	1559.50	110.0	1019.16	0.00%	113.4	3730.33	-3.05%	110.0	924.20	0.00%	120.1	3663.76	-9.19%
	26ch130	70.0	158.72	70.0	159.48	0.00%	70.0	496.45	0.00%	70.0	499.02	0.00%	70.0	2568.43	0.00%
	28pr136	53.0	39.16	53.0	37.48	0.00%	53.0	324.79	0.00%	53.0	32.23	0.00%	53.0	324.88	0.00%
	29pr144	60.0	616.99	60.0	875.78	0.00%	60.0	2448.41	0.00%	60.0	1535.54	0.00%	100.0	3724.15	-66.71%
	30ch150	61.0	703.97	61.0	549.95	0.00%	63.0	3744.96	-3.28%	61.0	570.16	0.00%	71.5	3732.29	-17.16%
	30kroA150	58.0	1061.34	58.0	1058.20	0.00%	58.0	1722.95	0.00%	58.0	617.23	0.00%	105.9	3803.33	-82.52%
	30kroB150	66.0	362.01	66.0	371.70	0.00%	66.0	2126.92	0.00%	66.0	343.15	0.00%	132.0	3634.76	-100.00%
	31pr152	57.0	877.35	57.0	2122.48	0.00%	105.0	3671.32	-84.21%	57.0	882.45	0.00%	105.0	3678.55	-84.21%
	32u159	76.0	1373.83	76.0	1267.41	0.00%	76.0	1784.42	0.00%	76.0	1296.21	0.00%	76.0	2652.70	0.00%
	39rat195	71.0	291.73	71.0	839.51	0.00%	71.0	2104.62	0.00%	71.0	287.79	0.00%	71.0	2021.79	0.00%
	40d198	70.0	85.51	70.0	101.60	0.00%	70.0	346.87	0.00%	70.0	85.84	0.00%	70.0	902.71	0.00%
	11berlin52	50.0	1.00	50.0	1.04	0.00%	50.0	0.59	0.00%	50.0	0.92	0.00%	50.0	1.07	0.00%
	11eil51	37.0	0.61	37.0	0.66	0.00%	37.0	0.56	0.00%	37.0	2.29	0.00%	37.0	0.72	0.00%
	14st70	56.0	0.78	56.0	0.77	0.00%	56.0	2.57	0.00%	56.0	0.76	0.00%	56.0	1.66	0.00%
	16eil76	51.0	7.95	51.0	8.11	0.00%	51.0	3.49	0.00%	51.0	2.70	0.00%	51.0	4.32	0.00%
	16pr76	70.0	135.46	70.0	146.88	0.00%	70.0	118.49	0.00%	70.0	129.47	0.00%	70.0	165.75	0.00%
	20kroA100	80.0	41.44	80.0	23.44	0.00%	80.0	1852.62	0.00%	80.0	40.97	0.00%	80.0	1038.06	0.00%
	20kroB100	86.0	50.14	86.0	71.47	0.00%	86.0	910.49	0.00%	86.0	50.21	0.00%	86.0	657.33	0.00%
	20kroC100	72.0	27.36	72.0	26.97	0.00%	72.0	137.27	0.00%	72.0	27.22	0.00%	72.0	116.40	0.00%
	20kroD100	78.0	10.11	78.0	10.13	0.00%	78.0	31.37	0.00%	78.0	9.97	0.00%	78.0	92.67	0.00%
	20kroE100	90.0	7.44	90.0	3.92	0.00%	90.0	255.75	0.00%	90.0	7.54	0.00%	90.0	68.45	0.00%
	20rat99	73.0	0.86	73.0	0.86	0.00%	73.0	6.50	0.00%	73.0	1.69	0.00%	73.0	0.67	0.00%
	20rd100	82.0	27.93	82.0	24.82	0.00%	82.0	138.71	0.00%	82.0	27.52	0.00%	82.0	57.48	0.00%
	21eil101	83.0	26.15	83.0	24.78	0.00%	83.0	33.38	0.00%	83.0	29.75	0.00%	83.0	49.03	0.00%
Set2	21lin105	95.0	344.37	95.0	361.56	0.00%	95.0	648.17	0.00%	95.0	329.39	0.00%	95.0	1106.10	0.00%
	22pr107	94.0	13.22	94.0	13.92	0.00%	94.0	12.05	0.00%	94.0	13.10	0.00%	94.0	8.95	0.00%
	25pr124	101.0	698.51	101.0	696.02	0.00%	101.0	3249.68	0.00%	101.0	697.18	0.00%	121.0	3619.24	-19.80%
	26bier127	126.0	3653.99	126.0	3659.72	0.00%	126.0	3649.08	0.00%	126.0	3654.25	0.00%	126.0	3640.60	0.00%
	26ch130	111.0	2392.90	111.0	2345.38	0.00%	129.0	3628.53	-16.22%	111.0	2190.43	0.00%	129.0	3629.11	-16.22%
	28pr136	120.0	36.70	120.0	35.96	0.00%	120.0	2716.97	0.00%	120.0	36.30	0.00%	120.0	3097.60	0.00%
	29pr144	143.0	3629.13	143.0	3629.15	0.00%	143.0	3636.00	0.00%	143.0	3629.55	0.00%	143.0	3629.63	0.00%
	30ch150	114.0	575.59	114.0	578.36	0.00%	149.0	3631.51	-30.70%	114.0	525.90	0.00%	149.0	3625.85	-30.70%
	30kroA150	149.0	3632.46	110.0	2324.17	26.17%	149.0	3622.59	0.00%	149.0	3631.89	0.00%	149.0	3633.49	0.00%
	30kroB150	145.0	3640.89	120.0	3108.51	17.22%	148.0	3630.07	-2.10%	144.7	3640.74	0.16%	148.0	3629.60	-2.10%
	31pr152	150.0	3629.36	150.0	3629.78	0.00%	150.0	3630.57	0.00%	150.0	3629.77	0.00%	150.0	3627.45	0.00%
	32u159	143.0	426.69	143.0	428.30	0.00%	154.0	3629.74	-7.69%	143.0	428.88	0.00%	154.0	3621.58	-7.69%
	39rat195	135.0	504.49	135.0	501.76	0.00%	135.0	324.21	0.00%	135.0	472.05	0.00%	135.0	539.33	0.00%
	40d198	149.0	2711.98	149.0	2713.38	0.00%	149.0	1501.18	0.00%	149.0	2728.54	0.00%	149.0	521.36	0.00%
AVG			**622.50**		**610.86**	**0.80%**		**1229.98**	**-2.73%**		**615.23**	**0.00%**		**1361.12**	**-8.08%**
#Worse						**0**			**7**			**0**			**11**

Table 12: Computational results for the five versions of the branch-and-cut algorithm on the instances with $\omega = 0.4$ and profit g_1.

		$\omega = 0.4$ and g_2														
		C-BC		**NoCond**			**NoCover**			**NoCluCover**			**NoPath**			
	Instance	**UB**	**Time**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	
	11berlin52	1829.0	0.59	1829.0	0.71	0.00%	1829.0	0.78	0.00%	1829.0	0.46	0.00%	1829.0	1.24	0.00%	
	11eil51	1279.0	0.57	1279.0	0.64	0.00%	1279.0	0.52	0.00%	1279.0	1.03	0.00%	1279.0	0.77	0.00%	
	14st70	1672.0	0.68	1672.0	0.66	0.00%	1672.0	0.81	0.00%	1672.0	0.48	0.00%	1672.0	0.88	0.00%	
	16eil76	2223.0	5.77	2223.0	2.71	0.00%	2223.0	6.60	0.00%	2223.0	1.74	0.00%	2223.0	8.12	0.00%	
	16pr76	2449.0	7.01	2449.0	8.01	0.00%	2449.0	11.52	0.00%	2449.0	3.75	0.00%	2449.0	7.97	0.00%	
	20kroA100	2151.0	38.03	2151.0	53.20	0.00%	2151.0	42.72	0.00%	2151.0	30.50	0.00%	2151.0	111.96	0.00%	
	20kroB100	2431.0	20.11	2431.0	19.48	0.00%	2431.0	11.65	0.00%	2431.0	15.81	0.00%	2431.0	43.78	0.00%	
	20kroC100	2174.0	30.16	2174.0	29.84	0.00%	2174.0	11.85	0.00%	2174.0	4.64	0.00%	2174.0	4.54	0.00%	
	20kroD100	1740.0	15.20	1740.0	15.15	0.00%	1740.0	21.36	0.00%	1740.0	7.84	0.00%	1740.0	13.49	0.00%	
	20kroE100	2415.0	2.75	2415.0	2.76	0.00%	2415.0	10.36	0.00%	2415.0	5.18	0.00%	2415.0	6.87	0.00%	
	20rat99	1905.0	1.24	1905.0	1.50	0.00%	1905.0	0.83	0.00%	1905.0	0.59	0.00%	1905.0	0.86	0.00%	
	20rd100	2228.0	15.07	2228.0	10.21	0.00%	2228.0	7.85	0.00%	2228.0	11.38	0.00%	2228.0	11.71	0.00%	
	21eil101	3365.0	34.06	3365.0	33.38	0.00%	3365.0	39.45	0.00%	3365.0	15.78	0.00%	3365.0	42.81	0.00%	
Set1	21lin105	2489.0	16.48	2489.0	26.24	0.00%	2489.0	14.38	0.00%	2489.0	12.89	0.00%	2489.0	21.65	0.00%	
	22pr107	2123.0	0.11	2123.0	0.11	0.00%	2123.0	0.06	0.00%	2123.0	0.10	0.00%	2123.0	0.09	0.00%	
	25pr124	2302.0	163.67	2302.0	167.95	0.00%	2302.0	2492.24	0.00%	2302.0	175.17	0.00%	4329.0	3628.61	-88.05%	
	26bier127	5420.0	2264.48	5420.0	1362.98	0.00%	5420.0	2988.36	0.00%	5420.0	2739.41	0.00%	5973.6	3746.84	-10.21%	
	26ch130	3423.0	1021.06	3423.0	495.95	0.00%	3423.0	2301.28	0.00%	3423.0	878.49	0.00%	3784.8	3729.98	-10.57%	
	28pr136	2699.0	106.06	2699.0	106.76	0.00%	2699.0	160.09	0.00%	2699.0	312.89	0.00%	2699.0	450.63	0.00%	
	29pr144	3055.0	2929.56	3055.0	2129.83	0.00%	3055.0	2643.36	0.00%	3055.0	1622.75	0.00%	5033.6	3753.32	-64.76%	
	30ch150	3131.0	817.59	3131.0	954.14	0.00%	3131.0	2304.69	0.00%	3131.0	514.97	0.00%	3131.0	3606.56	0.00%	
	30kroA150	3039.0	794.65	3039.0	840.56	0.00%	3039.0	1398.12	0.00%	3039.0	742.87	0.00%	4528.2	3741.84	-49.00%	
	30kroB150	3172.0	1848.50	3172.0	1672.67	0.00%	3705.3	3744.28	-16.81%	3172.0	1933.77	0.00%	5722.4	3772.42	-80.40%	
	31pr152	2915.0	1417.53	2915.0	1373.85	0.00%	5387.0	3649.19	-84.80%	2915.0	1433.18	0.00%	5387.0	3645.35	-84.80%	
	32u159	4002.0	517.02	4002.0	420.29	0.00%	4002.0	280.93	0.00%	4002.0	547.70	0.00%	4002.0	994.63	0.00%	
	39rat195	3656.0	454.65	3656.0	405.72	0.00%	3656.0	802.93	0.00%	3656.0	251.21	0.00%	3656.0	2780.70	0.00%	
	40d198	3595.0	129.76	3595.0	135.22	0.00%	3595.0	608.51	0.00%	3595.0	131.35	0.00%	3595.0	437.66	0.00%	
	11berlin52	2584.0	0.85	2584.0	0.95	0.00%	2584.0	0.63	0.00%	2584.0	0.86	0.00%	2584.0	0.77	0.00%	
	11eil51	1929.0	1.66	1929.0	1.60	0.00%	1929.0	2.39	0.00%	1929.0	0.59	0.00%	1929.0	0.65	0.00%	
	14st70	2736.0	1.50	2736.0	1.39	0.00%	2736.0	1.74	0.00%	2736.0	0.84	0.00%	2736.0	1.99	0.00%	
	16eil76	2518.0	32.22	2518.0	8.91	0.00%	2518.0	21.34	0.00%	2518.0	10.63	0.00%	2518.0	8.91	0.00%	
	16pr76	3550.0	32.22	3550.0	31.50	0.00%	3550.0	117.36	0.00%	3550.0	30.61	0.00%	3550.0	301.43	0.00%	
	20kroA100	3894.0	56.40	3894.0	63.12	0.00%	3894.0	641.03	0.00%	3894.0	54.43	0.00%	3894.0	1618.67	0.00%	
	20kroB100	4357.0	458.83	4357.0	393.65	0.00%	4357.0	620.17	0.00%	4357.0	394.08	0.00%	4357.0	1675.12	0.00%	
	20kroC100	3586.0	181.76	3586.0	104.91	0.00%	3586.0	366.21	0.00%	3586.0	95.66	0.00%	3586.0	222.14	0.00%	
	20kroD100	3799.0	56.77	3799.0	62.48	0.00%	3799.0	149.19	0.00%	3799.0	31.96	0.00%	3799.0	273.25	0.00%	
	20kroE100	4614.0	41.80	4614.0	10.00	0.00%	4614.0	38.30	0.00%	4614.0	27.59	0.00%	4614.0	43.07	0.00%	
	20rat99	3624.0	12.99	3624.0	17.58	0.00%	3624.0	21.41	0.00%	3624.0	42.50	0.00%	3624.0	2.94	0.00%	
	20rd100	4181.0	26.65	4181.0	32.46	0.00%	4181.0	79.52	0.00%	4181.0	28.32	0.00%	4181.0	42.35	0.00%	
	21eil101	4264.0	15.50	4264.0	46.97	0.00%	4264.0	56.60	0.00%	4264.0	45.93	0.00%	4264.0	74.82	0.00%	
Set2	21lin105	4814.0	362.29	4814.0	368.71	0.00%	4814.0	1682.85	0.00%	4814.0	339.21	0.00%	4814.0	411.31	0.00%	
	22pr107	4740.0	19.65	4740.0	16.97	0.00%	4740.0	11.05	0.00%	4740.0	18.69	0.00%	4740.0	52.44	0.00%	
	25pr124	6054.0	3624.39	5035.0	746.37	16.83%	6054.0	3623.37	0.00%	6054.0	3623.57	0.00%	6054.0	3623.35	0.00%	
	26bier127	6333.0	3635.77	6333.0	3633.48	0.00%	6333.0	3643.92	0.00%	6333.0	3636.61	0.00%	6333.0	3707.41	0.00%	
	26ch130	6393.0	3634.77	6393.0	3633.04	0.00%	6503.0	3625.46	-1.72%	6393.0	3635.65	0.00%	6503.0	3627.20	-1.72%	
	28pr136	6106.0	227.95	6106.0	228.75	0.00%	6106.0	1896.72	0.00%	6106.0	146.88	0.00%	6777.0	3624.38	-10.99%	
	29pr144	7242.0	3626.17	7242.0	3626.51	0.00%	7242.0	3626.72	0.00%	7242.0	3626.60	0.00%	7242.0	3630.24	0.00%	
	30ch150	6025.0	1209.80	6025.0	1214.47	0.00%	7533.0	3633.93	-25.03%	6025.0	995.18	0.00%	7533.0	3637.54	-25.03%	
	30kroA150	7456.0	3634.86	7456.0	3633.38	0.00%	7533.0	3626.17	-1.03%	7456.0	3634.65	0.00%	7533.0	3624.25	-1.03%	
	30kroB150	7524.0	3626.34	7524.0	3621.87	0.00%	7524.0	3627.64	0.00%	7524.0	3625.89	0.00%	7524.0	3628.93	0.00%	
	31pr152	7613.0	3630.61	7613.0	3628.96	0.00%	7613.0	3634.35	0.00%	7613.0	3631.17	0.00%	7613.0	3628.25	0.00%	
	32u159	7507.0	658.71	7507.0	435.99	0.00%	7850.0	3629.59	-4.57%	7507.0	919.53	0.00%	7860.0	3626.45	-4.70%	
	39rat195	6813.0	1805.72	6813.0	1419.77	0.00%	6813.0	857.97	0.00%	6813.0	292.75	0.00%	6813.0	598.08	0.00%	
	40d198	7480.0	378.10	7480.0	378.26	0.00%	9130.0	3637.92	-22.06%	7480.0	303.29	0.00%	7776.7	3648.54	-3.97%	
AVG			**808.27**		**696.90**	**0.31%**		**1230.15**	**-2.89%**		**751.66**	**0.00%**		**1479.63**	**-8.06%**	
#Worse						**0**			**7**			**0**			**13**	

Table 13: Computational results for the five versions of the branch-and-cut algorithm on the instances with $\omega = 0.4$ and profit g_2.

		$\omega = 0.6$ and g_1													
		C-BC		**NoCond**			**NoCover**			**NoCluCover**			**NoPath**		
	Instance	**UB**	**Time**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**
	11berlin52	43.0	3.91	43.0	3.86	0.00%	43.0	3.46	0.00%	43.0	3.88	0.00%	43.0	3.18	0.00%
	11eil51	39.0	1.75	39.0	1.96	0.00%	39.0	0.56	0.00%	39.0	1.81	0.00%	39.0	0.85	0.00%
	14st70	50.0	25.46	50.0	17.32	0.00%	50.0	28.17	0.00%	50.0	19.87	0.00%	50.0	37.84	0.00%
	16eil76	59.0	3.26	59.0	5.47	0.00%	59.0	26.91	0.00%	59.0	8.02	0.00%	59.0	19.06	0.00%
	16pr76	65.0	227.57	65.0	207.38	0.00%	65.0	487.74	0.00%	65.0	124.85	0.00%	65.0	2670.91	0.00%
	20kroA100	65.0	107.03	65.0	99.77	0.00%	65.0	2310.71	0.00%	65.0	106.66	0.00%	65.0	1640.40	0.00%
	20kroB100	66.0	98.45	66.0	99.49	0.00%	92.8	3629.78	-40.67%	66.0	97.08	0.00%	98.0	3629.25	-48.48%
Set1	20kroC100	62.0	74.74	62.0	95.02	0.00%	62.0	585.94	0.00%	62.0	72.43	0.00%	62.0	863.48	0.00%
	20kroD100	64.0	75.17	64.0	63.98	0.00%	64.0	2621.97	0.00%	64.0	75.22	0.00%	64.0	2162.33	0.00%
	20kroE100	63.0	174.86	63.0	175.40	0.00%	63.0	104.17	0.00%	63.0	166.56	0.00%	63.0	168.14	0.00%
	20rat99	52.0	49.10	52.0	58.21	0.00%	52.0	125.12	0.00%	52.0	46.19	0.00%	52.0	150.13	0.00%
	20rd100	72.0	211.81	72.0	209.05	0.00%	72.0	265.78	0.00%	72.0	368.21	0.00%	72.0	384.49	0.00%
	21eil101	82.0	91.61	82.0	123.22	0.00%	82.0	259.97	0.00%	82.0	82.72	0.00%	82.0	201.01	0.00%
	21lin105	78.0	248.31	78.0	342.03	0.00%	78.0	339.01	0.00%	78.0	132.16	0.00%	86.0	3635.17	-10.26%
	22pr107	77.0	3623.71	77.0	3623.48	0.00%	83.0	3626.10	-7.79%	77.0	3625.48	0.00%	83.0	3649.39	-7.79%
	11berlin52	51.0	0.15	51.0	0.15	0.00%	51.0	0.12	0.00%	51.0	0.14	0.00%	51.0	0.14	0.00%
	11eil51	50.0	3.74	50.0	3.88	0.00%	50.0	0.73	0.00%	50.0	3.76	0.00%	50.0	0.96	0.00%
	14st70	64.0	334.63	64.0	538.55	0.00%	64.0	1208.40	0.00%	64.0	300.55	0.00%	64.0	834.69	0.00%
	16eil76	74.0	176.04	74.0	621.02	0.00%	74.0	274.06	0.00%	74.0	175.62	0.00%	74.0	340.50	0.00%
	16pr76	74.0	1779.79	74.0	839.91	0.00%	75.0	3620.74	-1.35%	74.0	1777.29	0.00%	75.0	3619.31	-1.35%
	20kroA100	99.0	3623.43	99.0	3619.14	0.00%	99.0	3624.66	0.00%	99.0	3623.23	0.00%	99.0	3622.15	0.00%
	20kroB100	99.0	3621.47	99.0	3621.26	0.00%	99.0	3626.66	0.00%	99.0	3621.31	0.00%	99.0	3622.42	0.00%
Set2	20kroC100	99.0	3618.73	99.0	3619.69	0.00%	99.0	3625.41	0.00%	99.0	3618.66	0.00%	99.0	3627.18	0.00%
	20kroD100	99.0	3618.70	93.0	3039.27	6.06%	99.0	3621.29	0.00%	99.0	3618.94	0.00%	99.0	3624.66	0.00%
	20kroE100	97.0	2207.74	99.0	3616.08	-2.06%	99.0	3622.52	-2.06%	97.0	2215.67	0.00%	99.0	3618.93	-2.06%
	20rat99	87.0	188.38	87.0	223.63	0.00%	87.0	256.23	0.00%	87.0	194.56	0.00%	87.0	70.03	0.00%
	20rd100	99.0	2115.45	99.0	1210.77	0.00%	99.0	1885.67	0.00%	99.0	2135.44	0.00%	99.0	2752.34	0.00%
	21eil101	97.0	958.46	97.0	960.57	0.00%	100.0	3630.72	-3.09%	97.0	962.97	0.00%	100.0	3622.44	-3.09%
	21lin105	104.0	773.88	104.0	3070.16	0.00%	104.0	3638.71	0.00%	104.0	781.43	0.00%	104.0	3647.31	0.00%
	22pr107	106.0	11.02	106.0	11.22	0.00%	106.0	158.53	0.00%	106.0	11.02	0.00%	106.0	748.37	0.00%
AVG			**934.94**		**1004.03**	**0.13%**		**1573.66**	**-1.83%**		**932.39**	**0.00%**		**1765.57**	**-2.43%**
#Worse						**1**			**5**			**0**			**6**

		$\omega = 0.6$ and g_2													
		C-BC		**NoCond**			**NoCover**			**NoCluCover**			**NoPath**		
	Instance	**UB**	**Time**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**	**UB**	**Time**	**Gap%**
	11berlin52	2190.0	2.98	2190.0	2.82	0.00%	2190.0	2.17	0.00%	2190.0	3.72	0.00%	2190.0	3.70	0.00%
	11eil51	1911.0	4.63	1911.0	4.79	0.00%	1911.0	1.87	0.00%	1911.0	1.21	0.00%	1911.0	0.89	0.00%
	14st70	2589.0	25.40	2589.0	15.89	0.00%	2589.0	45.11	0.00%	2589.0	18.62	0.00%	2589.0	16.48	0.00%
	16eil76	3119.0	23.29	3119.0	23.06	0.00%	3119.0	82.51	0.00%	3119.0	19.60	0.00%	3119.0	46.33	0.00%
	16pr76	3275.0	56.14	3275.0	55.32	0.00%	3275.0	41.63	0.00%	3275.0	182.03	0.00%	3275.0	520.02	0.00%
	20kroA100	3192.0	219.32	3192.0	219.64	0.00%	3192.0	1467.65	0.00%	3192.0	135.17	0.00%	3192.0	2890.51	0.00%
	20kroB100	3203.0	161.54	3203.0	159.52	0.00%	3203.0	3506.17	0.00%	3203.0	165.41	0.00%	4753.0	3630.43	-48.39%
Set1	20kroC100	3110.0	503.42	3110.0	241.53	0.00%	3110.0	727.83	0.00%	3110.0	246.28	0.00%	3110.0	597.71	0.00%
	20kroD100	3133.0	80.11	3133.0	127.92	0.00%	3133.0	1584.18	0.00%	3133.0	81.68	0.00%	3133.0	2556.97	0.00%
	20kroE100	2950.0	136.87	2950.0	136.95	0.00%	2950.0	989.92	0.00%	2950.0	82.57	0.00%	2950.0	301.43	0.00%
	20rat99	2643.0	43.82	2643.0	50.08	0.00%	2643.0	67.76	0.00%	2643.0	41.06	0.00%	2643.0	131.01	0.00%
	20rd100	3591.0	390.73	3591.0	287.11	0.00%	3591.0	318.44	0.00%	3591.0	133.94	0.00%	3591.0	247.61	0.00%
	21eil101	4187.0	416.81	4187.0	326.02	0.00%	4187.0	358.10	0.00%	4187.0	412.53	0.00%	4187.0	644.74	0.00%
	21lin105	3955.0	181.35	3955.0	182.64	0.00%	3955.0	1324.81	0.00%	3955.0	162.10	0.00%	3955.0	1058.50	0.00%
	22pr107	3877.0	3626.58	4132.0	3627.09	-6.58%	4132.0	3625.61	-6.58%	3877.0	3626.40	0.00%	2697.0	3045.43	30.44%
	11berlin52	2608.0	0.15	2608.0	0.15	0.00%	2608.0	0.12	0.00%	2608.0	0.14	0.00%	2608.0	0.15	0.00%
	11eil51	2575.0	0.57	2575.0	0.56	0.00%	2575.0	0.88	0.00%	2575.0	0.56	0.00%	2575.0	1.79	0.00%
	14st70	3218.0	528.44	3218.0	334.69	0.00%	3218.0	2829.94	0.00%	3218.0	502.19	0.00%	3218.0	795.15	0.00%
	16eil76	3728.0	101.79	3728.0	87.29	0.00%	3728.0	441.42	0.00%	3728.0	100.66	0.00%	3728.0	92.92	0.00%
	16pr76	3729.0	476.85	3729.0	473.60	0.00%	3800.0	3623.70	-1.90%	3729.0	481.00	0.00%	3800.0	3623.80	-1.90%
	20kroA100	5008.0	3621.09	5008.0	3621.25	0.00%	5008.0	3627.00	0.00%	5008.0	3621.34	0.00%	5008.0	3626.83	0.00%
	20kroB100	5008.0	3622.99	5008.0	3622.28	0.00%	5008.0	3631.70	0.00%	5008.0	3623.16	0.00%	5008.0	3629.08	0.00%
Set2	20kroC100	5008.0	3618.73	5008.0	3618.58	0.00%	5008.0	3620.71	0.00%	5008.0	3619.27	0.00%	5008.0	3620.53	0.00%
	20kroD100	5008.0	3618.38	5008.0	3617.24	0.00%	5008.0	3622.52	0.00%	5008.0	3618.92	0.00%	5008.0	3630.76	0.00%
	20kroE100	5008.0	3617.79	4910.0	3423.90	1.96%	5008.0	3618.89	0.00%	5008.0	3617.95	0.00%	5008.0	3622.17	0.00%
	20rat99	4516.0	173.82	4516.0	220.51	0.00%	4516.0	123.91	0.00%	4516.0	154.69	0.00%	4516.0	72.77	0.00%
	20rd100	5008.0	3619.60	5008.0	1374.48	0.00%	5008.0	3028.85	0.00%	5008.0	3619.27	0.00%	5008.0	3620.90	0.00%
	21eil101	5050.0	3622.75	5037.0	3617.47	0.26%	5050.0	3621.34	0.00%	5050.0	3622.82	0.00%	5050.0	3628.80	0.00%
	21lin105	5228.0	3627.07	5228.0	3641.74	0.00%	5228.0	3641.79	0.00%	5228.0	3627.18	0.00%	5228.0	1629.92	0.00%
	22pr107	5363.0	132.19	5363.0	34.54	0.00%	5363.0	21.29	0.00%	5363.0	134.78	0.00%	5363.0	120.14	0.00%
AVG			**1208.51**		**1104.96**	**-0.15%**		**1653.26**	**-0.28%**		**1188.54**	**0.00%**		**1580.25**	**-0.66%**
#Worse						**1**			**2**			**0**			**2**

Table 14: Computational results for the five versions of the branch-and-cut algorithm on the instances with $\omega = 0.6$ and profits g_1 and g_2.

$\omega = 0.8$ and g_1

		C-BC		NoCond			NoCover			NoCluCover			NoPath		
	Instance	UB	Time	UB	Time	Gap%	UB	Time	Gap%	UB	Time	Gap%	UB	Time	Gap%
	11berlin52	47.0	6.35	47.0	8.06	0.00%	47.0	33.77	0.00%	47.0	6.36	0.00%	47.0	25.64	0.00%
	11eil51	43.0	2.21	43.0	5.67	0.00%	43.0	6.30	0.00%	43.0	2.10	0.00%	43.0	7.00	0.00%
	14st70	65.0	388.64	65.0	65.07	0.00%	65.0	2173.28	0.00%	65.0	361.26	0.00%	65.0	1216.32	0.00%
	16eil76	69.0	175.45	69.0	92.37	0.00%	69.0	668.67	0.00%	69.0	153.38	0.00%	69.0	741.48	0.00%
	16pr76	72.0	1645.28	72.0	1610.18	0.00%	75.0	3625.52	-4.17%	72.0	1653.24	0.00%	73.0	3619.85	-1.39%
	20kroA100	79.0	223.57	79.0	224.33	0.00%	99.0	3628.66	-25.32%	79.0	224.73	0.00%	99.0	3630.21	-25.32%
	20kroB100	86.0	2791.01	99.0	3642.47	-15.12%	99.0	3635.00	-15.12%	86.0	2802.72	0.00%	99.0	3637.24	-15.12%
Set1	20kroC100	83.0	448.35	83.0	394.61	0.00%	99.0	3630.59	-19.28%	83.0	417.48	0.00%	99.0	3630.37	-19.28%
	20kroD100	85.0	465.21	85.0	262.62	0.00%	99.0	3631.93	-16.47%	85.0	430.87	0.00%	99.0	3632.48	-16.47%
	20kroE100	80.0	334.23	80.0	182.45	0.00%	99.0	3625.67	-23.75%	80.0	347.91	0.00%	99.0	3625.78	-23.75%
	20rat99	79.0	1795.76	79.0	1587.52	0.00%	88.0	3624.21	-11.39%	79.0	1810.70	0.00%	88.0	3622.83	-11.39%
	20rd100	91.0	807.77	91.0	849.85	0.00%	99.0	3627.06	-8.79%	91.0	813.84	0.00%	99.0	3630.76	-8.79%
	21eil101	91.0	312.88	100.0	3628.12	-9.89%	100.0	3639.89	-9.89%	91.0	319.53	0.00%	100.0	3637.50	-9.89%
	21lin105	90.0	270.19	90.0	1541.15	0.00%	104.0	3639.01	-15.56%	90.0	289.91	0.00%	104.0	3644.62	-15.56%
	22pr107	106.0	3645.23	106.0	3642.42	0.00%	106.0	3643.92	0.00%	106.0	3645.25	0.00%	106.0	3632.86	0.00%
	11berlin52	51.0	0.03	51.0	0.03	0.00%	51.0	0.02	0.00%	51.0	0.03	0.00%	51.0	0.03	0.00%
	11eil51	50.0	0.84	50.0	0.51	0.00%	50.0	0.83	0.00%	50.0	0.79	0.00%	50.0	1.31	0.00%
	14st70	69.0	3.56	69.0	6.05	0.00%	69.0	134.31	0.00%	69.0	3.57	0.00%	69.0	17.66	0.00%
	16eil76	75.0	4.04	75.0	3.24	0.00%	75.0	13.35	0.00%	75.0	4.03	0.00%	75.0	2.65	0.00%
	16pr76	75.0	8.46	75.0	18.80	0.00%	75.0	3643.29	0.00%	75.0	8.15	0.00%	75.0	1583.03	0.00%
	20kroA100	99.0	10.19	99.0	27.53	0.00%	99.0	528.28	0.00%	99.0	9.73	0.00%	99.0	713.57	0.00%
	20kroB100	99.0	1202.32	99.0	3626.85	0.00%	99.0	3634.82	0.00%	99.0	1207.78	0.00%	99.0	3625.98	0.00%
Set2	20kroC100	99.0	3623.05	99.0	3622.40	0.00%	99.0	3631.01	0.00%	99.0	3623.28	0.00%	99.0	3631.17	0.00%
	20kroD100	99.0	3632.61	99.0	3634.61	0.00%	99.0	3629.19	0.00%	99.0	3631.99	0.00%	99.0	3622.21	0.00%
	20kroE100	99.0	3619.87	99.0	3626.20	0.00%	99.0	3637.93	0.00%	99.0	3620.10	0.00%	99.0	3630.18	0.00%
	20rat99	98.0	154.46	98.0	632.05	0.00%	98.0	939.68	0.00%	98.0	155.44	0.00%	98.0	1272.54	0.00%
	20rd100	99.0	135.40	99.0	98.07	0.00%	99.0	441.48	0.00%	99.0	135.97	0.00%	99.0	234.98	0.00%
	21eil101	100.0	1555.04	100.0	3625.01	0.00%	100.0	2144.42	0.00%	100.0	1569.38	0.00%	100.0	3625.32	0.00%
	21lin105	104.0	4.65	104.0	10.31	0.00%	104.0	3.26	0.00%	104.0	4.51	0.00%	104.0	6.79	0.00%
	22pr107	106.0	8.94	106.0	9.16	0.00%	106.0	12.09	0.00%	106.0	8.74	0.00%	106.0	16.52	0.00%
AVG			**909.19**		**1222.59**	**-0.83%**		**2174.25**	**-4.99%**		**908.76**	**0.00%**		**2130.63**	**-4.90%**
#Worse						**2**			**10**			**0**			**10**

$\omega = 0.8$ and g_2

		C-BC		NoCond			NoCover			NoCluCover			NoPath		
	Instance	UB	Time	UB	Time	Gap%	UB	Time	Gap%	UB	Time	Gap%	UB	Time	Gap%
	11berlin52	2384.0	10.57	2384.0	12.43	0.00%	2384.0	18.16	0.00%	2384.0	12.23	0.00%	2384.0	19.31	0.00%
	11eil51	2114.0	9.92	2114.0	12.42	0.00%	2114.0	9.15	0.00%	2114.0	6.94	0.00%	2114.0	7.18	0.00%
	14st70	3355.0	566.19	3355.0	134.57	0.00%	3355.0	659.94	0.00%	3355.0	533.94	0.00%	3355.0	372.47	0.00%
	16eil76	3573.0	60.10	3573.0	59.11	0.00%	3573.0	1294.30	0.00%	3573.0	85.04	0.00%	3573.0	2992.97	0.00%
	16pr76	3611.0	857.82	3611.0	843.61	0.00%	3694.0	3622.85	-2.30%	3694.0	3619.83	-2.30%	3765.0	3626.08	-4.26%
	20kroA100	4115.0	3088.95	4115.0	3015.71	0.00%	5008.0	3635.46	-21.70%	4115.0	3034.79	0.00%	5008.0	3632.68	-21.70%
	20kroB100	4916.0	3637.48	4916.0	3640.34	0.00%	5008.0	3628.18	-1.87%	4916.0	3637.40	0.00%	5008.0	3630.01	-1.87%
Set1	20kroC100	3999.0	304.44	3999.0	574.05	0.00%	5008.0	3626.03	-25.23%	3999.0	287.95	0.00%	5008.0	3624.21	-25.23%
	20kroD100	5008.0	3624.83	5008.0	3625.03	0.00%	5008.0	3621.51	0.00%	5008.0	3625.22	0.00%	5008.0	3629.87	0.00%
	20kroE100	4002.0	388.55	4002.0	384.94	0.00%	4703.0	3622.29	-17.52%	4002.0	388.90	0.00%	5008.0	3631.36	-25.14%
	20rat99	3992.0	3085.27	4297.0	3626.93	-7.64%	4434.0	3622.18	-11.07%	3992.0	2785.76	0.00%	4434.0	3625.47	-11.07%
	20rd100	5008.0	3627.15	5008.0	3629.51	0.00%	5008.0	3638.36	0.00%	5008.0	3626.99	0.00%	5008.0	3633.61	0.00%
	21eil101	4717.0	1873.07	4717.0	1878.69	0.00%	5050.0	3633.56	-7.06%	4717.0	1786.74	0.00%	5050.0	3626.43	-7.06%
	21lin105	5092.0	3638.94	4572.7	3664.68	10.20%	5228.0	3625.12	-2.67%	5092.0	3638.53	0.00%	5228.0	3646.06	-2.67%
	22pr107	5363.0	3636.41	5363.0	3630.58	0.00%	5363.0	3637.33	0.00%	5363.0	3636.09	0.00%	5363.0	3650.05	0.00%
	11berlin52	2608.0	0.08	2608.0	0.07	0.00%	2608.0	0.06	0.00%	2608.0	0.07	0.00%	2608.0	0.07	0.00%
	11eil51	2575.0	0.51	2575.0	0.50	0.00%	2575.0	0.48	0.00%	2575.0	0.51	0.00%	2575.0	0.62	0.00%
	14st70	3513.0	13.74	3513.0	13.62	0.00%	3513.0	24.55	0.00%	3513.0	13.25	0.00%	3513.0	81.31	0.00%
	16eil76	3800.0	163.65	3800.0	33.08	0.00%	3800.0	2.22	0.00%	3800.0	167.70	0.00%	3800.0	117.77	0.00%
	16pr76	3800.0	665.83	3800.0	248.03	0.00%	3800.0	984.88	0.00%	3800.0	670.13	0.00%	3800.0	677.96	0.00%
	20kroA100	5008.0	3624.63	5008.0	3138.14	0.00%	5008.0	3626.01	0.00%	5008.0	3623.81	0.00%	5008.0	3632.02	0.00%
	20kroB100	5008.0	3623.69	5008.0	3623.81	0.00%	5008.0	3629.08	0.00%	5008.0	3624.05	0.00%	5008.0	3636.64	0.00%
Set2	20kroC100	5008.0	3622.66	5008.0	3622.57	0.00%	5008.0	3643.09	0.00%	5008.0	3622.69	0.00%	5008.0	3636.16	0.00%
	20kroD100	5008.0	3635.24	5008.0	3633.28	0.00%	5008.0	3623.96	0.00%	5008.0	3634.34	0.00%	5008.0	3624.31	0.00%
	20kroE100	5008.0	3627.77	5008.0	3627.91	0.00%	5008.0	3630.97	0.00%	5008.0	3627.43	0.00%	5008.0	3639.56	0.00%
	20rat99	5007.0	324.77	5007.0	110.66	0.00%	5007.0	209.81	0.00%	5007.0	323.20	0.00%	5007.0	1748.72	0.00%
	20rd100	5008.0	43.69	5008.0	72.08	0.00%	5008.0	10.58	0.00%	5008.0	45.00	0.00%	5008.0	10.01	0.00%
	21eil101	5050.0	3631.53	5050.0	3626.64	0.00%	5050.0	3630.00	0.00%	5050.0	3629.89	0.00%	5050.0	3643.53	0.00%
	21lin105	5228.0	1532.00	5228.0	1827.56	0.00%	5228.0	3518.19	0.00%	5228.0	1541.45	0.00%	5228.0	2311.88	0.00%
	22pr107	5363.0	137.27	5363.0	33.96	0.00%	5363.0	280.41	0.00%	5363.0	138.40	0.00%	5363.0	774.23	0.00%
AVG			**1768.56**		**1744.82**	**0.09%**		**2290.29**	**-2.98%**		**1845.61**	**-0.08%**		**2362.75**	**-3.30%**
#Worse						**1**			**8**			**1**			**8**

Table 15: Computational results for the five versions of the branch-and-cut algorithm on the instances with $\omega = 0.8$ and profits g_1 and g_2.