Archetti, C.; Corberán, A.; Plana, I.; Sanchís Llopis, JM.; Speranza, M. (2016). A branch-and-cut algorithm for the Orienteering Arc Routing Problem. Computers & Operations Research. 66:95-104. https://doi.org/10.1016/j.cor.2015.08.003

A branch-and-cut algorithm for the Orienteering Arc Routing Problem

Claudia Archetti[(1)] Ángel Corberán[(2)*] Isaac Plana [(3)]
José M. Sanchis [(4)] M. Grazia Speranza[(1)]

[(1)] *Dipartimento di Economia e Management, Universitá di Brescia, Italy*
[(2)] *Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain*
[(3)] *Departamento Matemáticas para la Economía y la Empresa, Universidad de Valencia, Spain*
[(4)] *Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Spain*

{claudia.archetti, grazia.speranza}@unibs.it, {angel.corberan, isaac.plana}@uv.es, jmsanchis@mat.upv.es

December 3, 2014

Abstract

In arc routing problems, customers are located on arcs, and routes of minimum cost have to be identified. In the Orienteering Arc Routing Problem (OARP), in addition to a set of regular customers that have to be serviced, a set of potential customers is available. From this latter set, customers have to be chosen on the basis of an associated profit. The objective is to find a route servicing the customers which maximize the total profit collected while satisfying a given time limit on the route. In this paper, we describe large families of facet-inducing inequalities for the OARP and present a branch-and-cut algorithm for its solution. The exact algorithm embeds a procedure which builds a heuristic solution to the OARP on the basis of the information provided by the solution of the linear relaxation. Extensive computational experiments over different sets of OARP instances show that the exact algorithm is capable of solving large instances, with up to 1500 vertices and 10500 arcs, within one hour and often within a few minutes.

Keywords: Orienteering Arc Routing Problem, Routing Problems with Profits, Branch-and-Cut.

1 Introduction

Arc routing problems deal with the design of routes traversing a subset of arcs or edges of a graph. The arcs or edges that have to be traversed in an arc routing problem are usually called *customers*. In the classical arc routing setting, all customers must be serviced and the objective is to minimize the total cost of the routes. A new class of arc routing problems is gaining attention in the last years which is called *arc routing problems with profits*. In problems of this class, the given set of customers comprises a subset of *required customers*,

*corresponding author

i.e., customers that have to be serviced, and a subset of *profitable customers*. For each profitable customer a decision has to be taken to determine whether it will be serviced or not. A profit is associated with each profitable customer and the decision concerning their service is based on convenience.

As it happens for the classical routing problems, that is routing problems without profitable customers, the literature on *vehicle routing problems with profits*, where customers are placed on nodes of a graph, is much broader than the one on their arc routing counterpart, where customers are placed on arcs or edges. We refer the reader to Archetti, Speranza and Vigo [6] and Archetti and Speranza [4] for two recent surveys of vehicle and arc routing problems with profits, respectively.

Routing problems with profits deal with the dichotomy of maximizing the profit collected and minimizing the traveling cost. These two contrasting objectives can be handled in different ways and this gives rise to different types of problems. Vehicle routing problems with profits can be classified in three classes (see Feillet, Dejax, and Gendreau [8] and Archetti, Speranza and Vigo [6]):

- **Prize-collecting problems**: a lower bound is set on the total profit collected and the objective is to minimize the traveling cost;
- **Profitable problems**: the objective is to maximize the difference between the collected profit and the traveling cost and no bound is fixed neither on traveling cost nor on profit collected;
- **Orienteering problems**: an upper bound is set on the traveling cost and the collected profit is maximized.

In orienteering problems the traveling cost is often interpreted as traveling time and the upper bound as a limit on the time duration of a route.

Adapting the above classification to the arc routing problems, we define the Orienteering Arc Routing Problem (OARP) as follows. Customers are placed on arcs or edges of a graph. Profitable and required customers, i.e. customers which must be serviced and have no associated profit, are given. Note that in the Orienteering Problem, the node routing counterpart of the OARP, required customers are typically not considered. The OARP aims at finding a single route which maximizes the total collected profit while satisfying a maximum time duration.

The OARP can model a number of real-life applications. For instance, nowadays it is more and more frequent that demand for transportation services is posted on the web and carriers respond to that demand and offer their services, possibly in the framework of an electronic auction. Within this set of potential customers, the carrier has to select those which best fit for its fleet and its customers. In an electronic auction, the carrier will make a bid for these customers. In a truck-load type of service, a transportation service consists of reaching a node with an empty truck, filling the truck with a load, traversing an arc and unloading the truck completely. A vehicle (or fleet of vehicles) with limited traveling time is usually available to perform the service. The carrier may need to take into account a set of regular customers who have to be serviced. This is, in fact, the most common situation. The carrier looks for additional profitable customers in order to fully use the unused traveling time of the vehicle. The above situation can be modeled as an arc routing problem with profits. Consider a set of customers as the one depicted in Figure 1(a), where required and profitable customers are represented by dotted and solid arcs, respectively. A profit is associated with

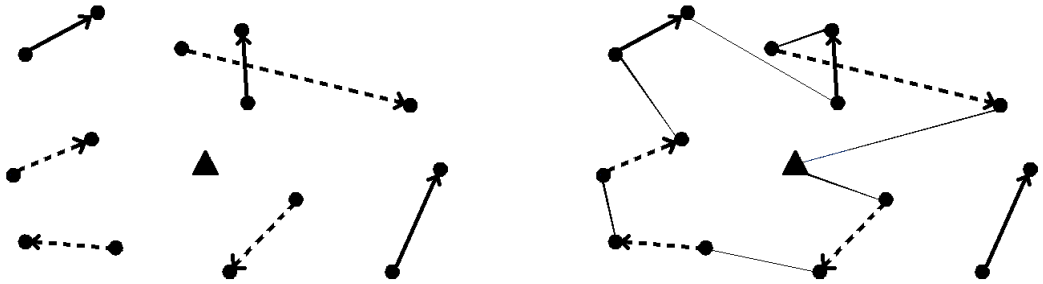

Figure 1: OARP example.

each profitable customer, a traveling time is associated with each arc of the graph, and a maximum traveling time is available to perform the service. The objective is to find a route leaving from and coming back to the depot (represented as a triangle in Figure 1), with total time duration not greater than the maximum traveling time, that maximizes the profit of the profitable customers that are serviced (see Figure 1(b)).

To the best of our knowledge, the only paper that is closely related to the OARP is due to Souffriau et al. [10]. In this paper the authors consider a practical application where cycle trips for tourists have to be built in the province of East Flanders. The differences with respect to the OARP we have defined are that no required customers are considered and a profit is associated with every arc of the graph. The authors propose a mathematical formulation, which assumes that the graph is complete, and a fast heuristic algorithm based on the greedy randomized adaptive search scheme. Computational tests are performed on instances generated from data related to the real application.

The multiple vehicle version of the OARP, called the *Team Orienteering Arc Routing Problem* (TOARP), has been studied in Archetti et al. [3, 2]. In [3], a formulation for the problem is presented, its polyhedron is studied, and a branch-and-cut algorithm solving small and medium size instances is described. In [2] a matheuristic is proposed.

In this paper we study the OARP and propose an exact algorithm for its solution. Furthermore, we present a heuristic which makes use of the information provided by the linear relaxation and builds a feasible solution to the OARP through the exact solution of a minimum cost network flow problem. The heuristic is embedded into the exact branch-and-cut algorithm and provides a good lower bound that helps pruning the nodes of the branch-and-bound tree. This algorithm can be classified as a *matheuristic* as it embeds the exact solution of a Mixed Integer Linear Programming model (MILP). Matheuristics are more and more used for the solution of hard combinatorial problems in general and for routing problems in particular, as witnessed by the large amount of contributions surveyed in Archetti and Speranza [5]. Extensive computational experiments over different sets of OARP instances show that the exact algorithm is capable of solving large instances, with up to 1500 vertices and 10500 arcs, within one hour and often within a few minutes.

The paper is organized as follows. In Section 2 we define the problem and present a mathematical programming formulation. The associated polyhedron and several families of valid inequalities are described in Section 3. The proposed branch-and-cut algorithm is presented in Section 4, while the computational experiments are described in Section 5. Finally, some conclusions are drawn in Section 6.

2 Problem definition and formulation

Let $G = (V, A)$ be a directed graph, where vertex 1 represents the depot. Let $A_R \subseteq A$ be the set of required arcs, those that have to be serviced mandatorily, and let $A_P \subseteq A$ be the set of profitable arcs, those that may be serviced if beneficial. Each arc $(i, j) \in A_P$ is associated with a nonnegative profit s_{ij} that can be collected at most once, and each arc $(i, j) \in A$ is associated with a traveling time c_{ij}. The OARP consists in finding a route, starting and ending at the depot, such that its total time duration is not greater than T_{max}, all the arcs in A_R are traversed at least once, and the sum of the profits of the traversed arcs in A_P is maximum. The traveling time on every traversed arc contributes to the time duration of the route.

As in most arc routing problems, we assume that all the vertices are incident with arcs in $A_R \cup A_P$. When G does not satisfy this condition, it can be transformed into an equivalent graph which does. This assumption makes the formulation and the solution of the problem easier.

We use the following notation. Given a subset of vertices $S \subseteq V$, let $A(S)$ be the set of arcs with both endpoints in S, $A(S) = \{(i, j) \in A : i, j \in S\}$. We define similar sets $A_P(S)$ and $A_R(S)$. The subgraph of G induced by the set of vertices S is denoted by $G(S)$. Given two disjoint sets $S, T \subseteq V$, we define $(S, T) = \{(i, j) \in A : i \in S, j \in T\}$, $(S : T) = (S, T) \cup (T, S)$ and $\delta(S) = (S : V \setminus S)$. Finally, given a set of arcs $W \subset A$ and a vector x indexed by the arcs in A, $x(W) = \sum_{a \in W} x_a$.

We formulate the OARP by using the following variables:

- For each $(i, j) \in A$, let x_{ij} be the number of times arc (i, j) is traversed.
- For each $(i, j) \in A_R \cup A_P$, let y_{ij} be a binary variable that takes value 1 if arc (i, j) is serviced and 0 otherwise.

The OARP can be formulated as follows:

$$\text{Maximize} \quad \sum_{(i,j) \in A_P} s_{ij}\, y_{ij}$$

$$\sum_{j \in V \setminus \{i\}} x_{ij} = \sum_{j \in V \setminus \{i\}} x_{ji} \quad \forall i \in V \tag{1}$$

$$\sum_{i \in V \setminus S,\ j \in S} x_{ij} \geq 1 \quad \forall S \subset V \setminus \{1\} \quad \text{with} \quad A_R(S) \neq \emptyset \tag{2}$$

$$\sum_{i \in V \setminus S,\ j \in S} x_{ij} \geq y_a \quad \forall S \subset V \setminus \{1\},\ \ \forall a \in A_P(S) \tag{2'}$$

$$x_{ij} \geq 1 \quad \forall (i, j) \in A_R \tag{3}$$

$$x_{ij} \geq y_{ij} \quad \forall (i, j) \in A_P \tag{3'}$$

$$\sum_{(i,j)\in A} c_{ij}x_{ij} \leq T_{max} \qquad (4)$$

$$x_{ij} \geq 0 \text{ and integer} \quad \forall (i,j) \in A \qquad (5)$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A_P, \qquad (6)$$

where (1) are the symmetry equations, connectivity constraints (2) and (2') ensure that the route is connected to the depot, constraints (3) and (3') force the vehicle to traverse the arcs that it services, and constraint (4) limits the traveling time of the route.

3 The OARP polyhedron

Determining the dimension of the polyhedron defined as the convex hull of the OARP solutions is a very difficult task, because it does not only depend on the number of vertices and arcs, but also on the traveling times on the arcs and the time limit T_{max}. The difficulty in obtaining the dimension makes the task of proving that an inequality is facet inducing almost impossible in general. In Archetti et al. [3], the constraints limiting the duration of the routes were removed from the polyhedron of the TOARP. We proceed in the same way here.

If we remove constraint (4), given that the OARP is a special case of the TOARP, the convex hull of the solutions satisfying (1) to (3'), (5) and (6) is a polyhedron of dimension $(|A| + |A_P| - |V| + 1)$ (if G is strongly connected) and the following inequalities are facet inducing:

- $x_{ij} \geq 0$, for all $(i,j) \in A \setminus (A_R \cup A_P)$, if $G \setminus \{(i,j)\}$ is strongly connected. If $(i,j) \in A_R \cup A_P$, then $x_{ij} \geq 0$ is not facet-inducing because it is dominated by the corresponding inequality $x_{ij} \geq y_{ij}$;
- $y_{ij} \geq 0$, for all $(i,j) \in A_R \cup A_P$, if $G \setminus \{(i,j)\}$ is strongly connected. Inequalities $y_{ij} \leq 1$ are not facet-inducing because they are dominated by inequalities $y_{ij} \geq 0$ and inequalities (3) or (3'), if $(i,j) \in A_R$ or $(i,j) \in A_P$, respectively;
- $x_{ij} \geq y_{ij}$, for all $(i,j) \in A_P$, if $G \setminus \{(i,j)\}$ is strongly connected;
- $x_{ij} \geq 1$, for all $(i,j) \in A_R$, if $G \setminus \{(i,j)\}$ is strongly connected;
- $\sum_{i\in V\setminus S, j\in S} x_{ij} \geq 1$, for all $S \subset V\setminus\{1\}$, for all $a \in A_R(S)$, if subgraphs $G(S)$ and $G(V \setminus S)$ are strongly connected;
- $\sum_{i\in V\setminus S, j\in S} x_{ij} \geq y_a$, for all $S \subset V\setminus\{1\}$, for all $a \in A_P(S)$, if subgraphs $G(S)$ and $G(V \setminus S)$ are strongly connected.

In addition, several families of valid inequalities for the TOARP were described in Archetti et al. [3], namely the K-C, path-bridge, and max-length inequalities. In what follows, we present the version of these inequalities for the OARP. Although the new inequalities can not be directly obtained from the results presented in [3], similar proofs show that they are valid for the OARP. For this reason, we omit the proofs.

3.1 K-C inequalities

K-C inequalities are a well-known family of facet-inducing inequalities for different variants of arc routing problems.

A K-C inequality is defined by a partition of V into $K+1$ sets $\{M_0, M_1, M_2, \ldots, M_{K-1}, M_K\}$, with $K \geq 3$, where all the required arcs are contained either in $A(M_0 : M_K)$ or in a set $A(M_i)$. Let $F \subseteq (M_0 : M_K)_P$ be a set of profitable arcs such that $|F| + |(M_0 : M_K)_R|$ is an even number greater than or equal to 2. Let I_R be the set of indices $i \in \{1, \ldots, K-1\}$ such that either $A(M_i)$ contains a required arc or M_i contains the depot:

$$I_R = \{i \ : \ A_R(M_i) \neq \emptyset \ \text{ or } \ 1 \in M_i\}.$$

For each of the remaining indices $i \notin I_R$ we assume there is a profitable arc $a_i \in A(M_i)$. Let us denote $H = \{a_i \ : \ i \notin I_R\}$. We call *K-C inequality*:

$$\sum_{(i,j)\in A} b_{ij}\, x_{ij} \geq (K-2)|(M_0 : M_K)_R| + (K-2)(2y(F) - |F|) + 2|I_R| + 2y(H), \qquad (7)$$

where the coefficients b_{ij} are shown in Figure 2. Each number close to an arc represents the coefficient of the variable x_{ij} associated with the traversal of the corresponding arc:

$$b_{ij} = \begin{cases} K-2 & \text{if } (i,j) \in (M_0 : M_K) \\ |r-s| & \text{if } (i,j) \in (M_r : M_s),\ \{r,s\} \neq \{0,K\} \\ 0 & \text{otherwise.} \end{cases}$$

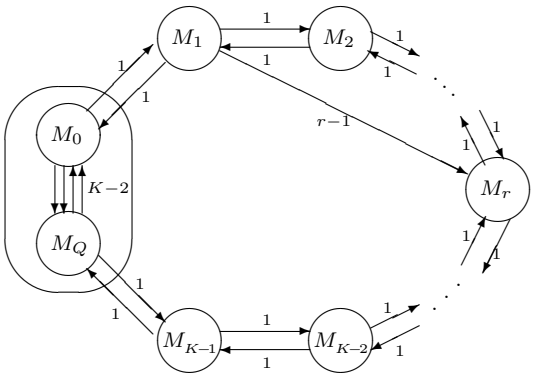

Figure 2: K-C configuration

3.2 2-PB inequalities

Path-Bridge (PB) inequalities were introduced by Letchford [9] for the undirected General Routing Problem (GRP) and extended to the TOARP in Archetti et al. [3]. They are also valid for the OARP, but since the separation of the general case is difficult, we only consider the particular case with two paths, called 2-PB inequalities, which is the one that we describe in what follows.

Consider a partition $\left\{M_A, M_Z, \{M_q^1\}_{q=1,\ldots,n_1}, \{M_q^2\}_{q=1,\ldots,n_2}\right\}$ of V, where n_1 and n_2 are integer numbers greater than or equal to 2, and all the required arcs are contained either in

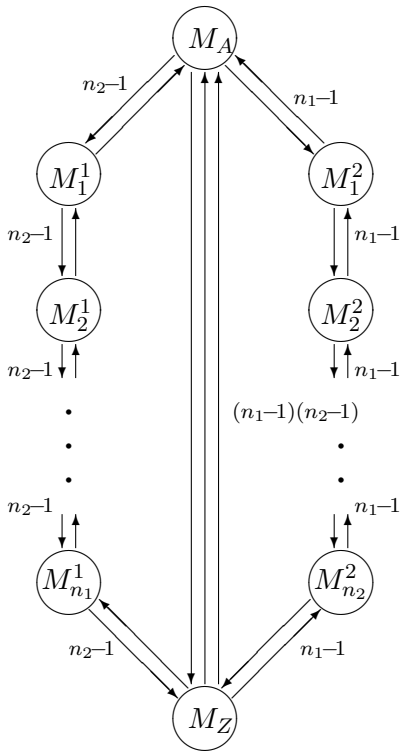

Figure 3: 2-PB configuration.

$M_A \cup M_Z$ or in a set M_q^s, $s = 1, 2$. For the sake of simplicity, we identify M_A with M_0^s and M_Z with $M_{n_s+1}^s$, for both $s = 1$ and $s = 2$. Let $F \subseteq (M_A : M_Z)_P$ be a set of profitable arcs such that $|F| + |(M_A : M_Z)_R|$ is an odd number. Hence, a 2-PB inequality is defined by 2 *paths* from M_A to M_Z, each of them with $n_s + 2$ node-sets, and by $|F| + |(M_A : M_Z)_R|$ required and profitable arcs joining the sets M_A and M_Z, which form the *bridge*. For each path s, let I_R^s be the set of indices $i \in \{1, \ldots, n_s\}$ such that either $A(M_i^s)$ contains a required arc or M_i^s contains the depot:

$$I_R^s = \{i \ : \ \ A_R(M_i^s) \neq \emptyset \ \text{ or } \ 1 \in M_i^s\}.$$

For each of the remaining indices in path s, $i \notin I_R^s$, we assume there is a profitable arc $a_i^s \in A(M_i^s)$ and we denote $H^s = \{a_i^s \ : \ \ i \notin I_R^s\}$.

The coefficients of the 2-PB inequality are $b_{ij} = c(M_q^s, M_r^t)$, where:

- $c(M_A, M_Z) = c(M_Z, M_A) = 1$,
- $c(M_q^s, M_r^s) = \frac{|q-r|}{n_s - 1}$ for all $q, r \in \{0, 1, \ldots, n_{s+1}\}$, $s \in \{1, 2\}$,
- $c(M_q^1, M_r^2) = c(M_r^2, M_q^1) = \frac{1}{n_1 - 1} + \frac{1}{n_2 - 1} + \left|\frac{q-1}{n_1-1} - \frac{r-1}{n_2-1}\right| \ \ q \in \{1, \ldots, n_1\}, r \in \{1, \ldots, n_2\}$,

and $b_{ij} = 0$ otherwise. The *2-PB inequality* is then

$$\sum_{(i,j) \in A} b_{ij}\, x_{ij} \geq \Big|(M_A : M_Z)_R\Big| + \Big(2y(F) - |F|\Big) + \sum_{s=1}^{2} \frac{2}{n_s - 1} \Big(|I_R^s| + y(H_s)\Big) - 1. \qquad (8)$$

Note that, if we multiply the above inequality by $(n_1-1)(n_2-1)$, all the coefficients can be expressed with integer numbers and the inequality can be rewritten as:

$$(n_1-1)(n_2-1)\sum_{(i,j)\in A} b_{ij}\,x_{ij} \geq (n_1-1)(n_2-1)\Big(\Big|(M_A:M_Z)_R\Big| + 2y(F) - |F|\Big)$$

$$+2(n_2-1)\Big(|I_R^1| + y(H_1)\Big) + 2(n_1-1)\Big(|I_R^2| + y(H_2)\Big) - 1. \tag{9}$$

This configuration is shown in Figure 3, where the number close to each pair of arcs joining sets M_q^s and M_r^t represents the coefficient b_{ij} associated with the variables of the arcs $(i,j) \in (M_q^s, M_r^t)$.

3.3 Max-time constraints

This class of valid inequalities is related to the maximum duration of the route.

Let F be a subset of profitable arcs. Consider the problem of finding a tour of minimum duration traversing all the arcs in $F \cup A_R$ and visiting the depot. This problem is known as the Directed General Routing Problem (DGRP) and has been recently studied in Ávila et al. [7], where a branch-and-cut algorithm producing good computational results is described. Let $z(F \cup A_R)$ represent the optimal value of the DGRP on the graph induced by $F \cup A_R$. If $z(F \cup A_R) > T_{max}$, then the vehicle cannot service all the arcs in F and hence the inequality

$$y(F) \leq |F| - 1 \tag{10}$$

is valid.

3.4 Cover inequalities

Let us suppose that the Linear Programming (LP) model solved at the root node has an optimal value $z(LP_0)$. This value is an upper bound on the optimal OARP value and, hence, any set of profitable arcs $\{a_1, \ldots, a_Q\}$ such that the sum of corresponding profits exceeds $z(LP_0)$ corresponds to an infeasible solution. Then, any OARP solution must satisfy the following inequality:

$$y_{a_1} + \cdots + y_{a_Q} \leq |Q|. \tag{11}$$

For instance, consider an optimal LP solution at the root node like the one depicted in Figure 4, where again required and profitable customers are represented by dotted and solid arcs, respectively. The optimal value is $z(LP_0) = 34$, while the sum of the profits of the profitable arcs serviced by the vehicle is 40. Therefore, we have the following cover inequality:

$$y_1 + y_2 + y_3 + y_4 \leq 3,$$

which is violated by this solution. Note that if we consider the LP solution at any other node k of the branch-and-cut tree, the corresponding inequality is not valid for any OARP solution, but it must be satisfied by all the solutions of the subproblems associated with the children of node k, and it can therefore be used as a local cut.

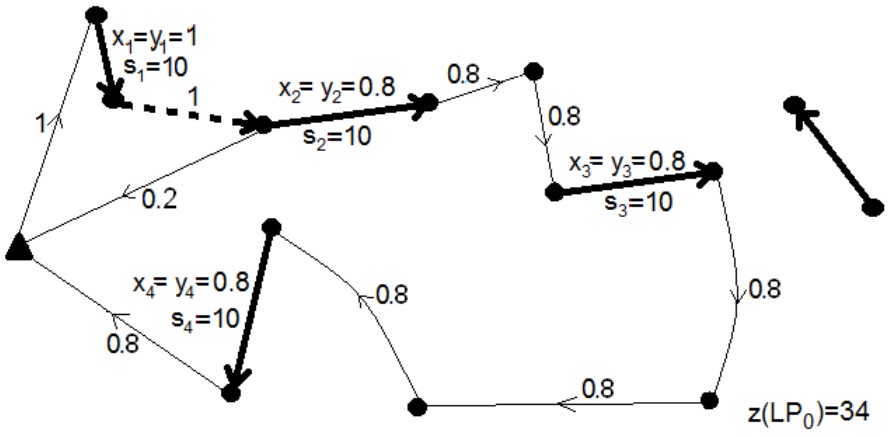

Figure 4: Solution violating a cover inequality.

4 The branch-and-cut algorithm

In this section we present a branch-and-cut algorithm that incorporates separation procedures for the inequalities described in the previous section as well as a lower bound obtained through a heuristic which makes use of the information provided by the linear relaxation of formulation (1)-(6).

4.1 A heuristic algorithm

To obtain lower bounds that help prune the branch-and-cut tree, we designed a heuristic that, starting from an optimal solution of any LP relaxation, produces a feasible solution to the OARP. Such heuristic makes use of the information provided by the linear relaxation of a mathematical programming formulation and can be classified as a matheuristic (see Archetti and Speranza [5] for a recent survey on matheuristics for routing problems).

We illustrate the heuristic by means of an example. Consider the fractional LP solution in Table 1 obtained at a certain node of the branch-and-cut tree. Note that most of the x variables take integer values and only a few of them are fractional. Let us consider the weighted graph induced by the arcs associated with the variables with fractional values, and define their weights as the fractional parts of the LP solution (see Figure 5.a). Note that all the vertices are balanced except for vertex 49, which can be considered a source node with supply 1, and vertex 25, which can be considered a sink node with demand 1 (see Figure 5.b). If we solve the minimum cost flow problem from the source node to the sink node on the original graph G, we can replace the fractional part of the LP solution by that of the flow problem, thus obtaining an integer solution.

The general scheme of the heuristic is the following:

1. Consider the subgraph G' induced by the arcs which have a positive fractional part in the solution of the linear relaxation;

   (a) Assign a weight w_{ij} to each arc of G' equal to the fractional part of the corresponding x variable;

(b) Identify in G' the set of source nodes having a positive balance and the set of sink nodes having a negative balance;

2. Solve a minimum cost network flow on G with the source and sink nodes defined in 1(b).

3. Replace the fractional part of the LP solution by that of the flow problem.

Not all the solutions obtained with the algorithm described above are feasible solutions to the OARP. They may be infeasible because their duration may exceed T_{max} or because they induce a disconnected graph. This happens mainly when the number of unbalanced vertices is high. For this reason, we run this procedure only when the number of unbalanced vertices is less than 10. Note that disconnected solutions could be easily modified and made connected. However, some computational tests have shown that the quality of the solutions obtained in this way is not good and, thus, we simply discard disconnected solutions.

This heuristic has resulted to be quite successful in finding good solutions. Preliminary tests have shown that, however, it is too cumbersome to run the heuristic at every node of the branch-and-cut tree. Thus, we execute the heuristic at the first 10 nodes. After that and until node 50, it is executed only once every two nodes. From node 50 on, it is executed only once every 10 nodes.

x(2,37) = 1	y(2,37) = 1	x(39,43) = 1	y(39,43) = 1	x(1,12) = 1
x(4,14) = 1.23	y(4,14) = 1	x(2,3) = 1	y(2,3) = 1	x(3,2) = 1
x(6,37) = 1	y(6,37) = 1	x(9,20) = 1	y(9,20) = 1	x(5,11) = 1
x(8,7) = 1	y(8,7) = 1	x(11,5) = 1	y(11,5) = 1	x(7,33) = 2
x(14,23) = 1	y(14,23) = 1	x(13,38) = 1	y(13,38) = 1	x(10,26) = 1
x(15,27) = 1	y(15,27) = 1	x(16,29) = 1	y(16,29) = 1	x(15,44) = 1
x(21,39) = 1	y(21,39) = 1	x(18,28) = 1	y(18,28) = 1	x(18,25) = 1.77
x(22,43) = 1	y(22,43) = 1	x(19,36) = 1	y(19,36) = 1	x(27,40) = 2
x(24,37) = 1	y(24,37) = 1	x(19,1) = 1	y(19,1) = 1	x(29,16) = 1
x(26,38) = 1	y(26,38) = 1	x(20,18) = 2	y(20,18) = 1	x(30,38) = 1
x(26,34) = 1	y(26,34) = 1	x(34,30) = 1	y(34,30) = 1	x(32,45) = 1
x(31,43) = 1	y(31,43) = 1	x(40,8) = 1	y(40,8) = 1	x(35,47) = 1
x(33,7) = 1	y(33,7) = 1	x(42,50) = 1	y(42,50) = 1	x(36,19) = 1
x(41,9) = 1	y(41,9) = 1	x(44,27) = 1	y(44,27) = 1	x(37,2) = 2
x(43,41) = 1	y(43,41) = 1	x(45,5) = 1	y(45,5) = 1	x(38,26) = 2
x(46,48) = 1	y(46,48) = 1	x(50,6) = 1	y(50,6) = 1	x(39,21) = 1
x(18,49) = 1	y(18,49) = 1	x(3,17) = 1	y(3,17) = 1	x(43,31) = 1
x(9,29) = 1	y(9,29) = 1	x(25,20) = 1	y(25,20) = 1	x(43,22) = 1
x(38,30) = 1	y(38,30) = 1	x(44,10) = 1	y(44,10) = 1	x(48,44) = 1
x(26,13) = 1	y(26,13) = 1	x(47,7) = 1	y(47,7) = 1	x(49,18) = 1.77
x(23,1) = 1	y(23,1) = 1	x(2,39) = 1	y(2,39) = 1	x(49,28) = 0.23
x(35,24) = 1	y(35,24) = 1	x(40,15) = 1	y(40,15) = 1	x(29,46) = 1
x(45,19) = 1	y(45,19) = 1	x(14,25) = 0.23	y(14,25) = 0.23	x(1,6) = 1
x(7,35) = 1	y(7,35) = 1	x(33,35) = 1	y(33,35) = 1	x(5,45) = 1
x(26,46) = 1	y(26,46) = 1	x(30,10) = 1	y(30,10) = 1	x(17,14) = 1
x(46,15) = 1	y(46,15) = 1	x(25,9) = 1	y(25,9) = 1	x(28,4) = 1.23
x(6,32) = 1	y(6,32) = 1	x(44,26) = 1	y(44,26) = 1	x(10,44) = 1
x(12,3) = 1	y(12,3) = 1	x(14,49) = 1	y(14,49) = 1	x(37,42) = 1

Table 1: LP solution. Total traveling time: 13123 T_{max}: 13123

4.2 Cutting-plane procedure

The initial LP relaxation contains symmetry equations (1), traversing inequalities (3) and (3'), inequalities (4) limiting the duration of the route, trivial inequalities ($x_{ij} \geq 0$ and

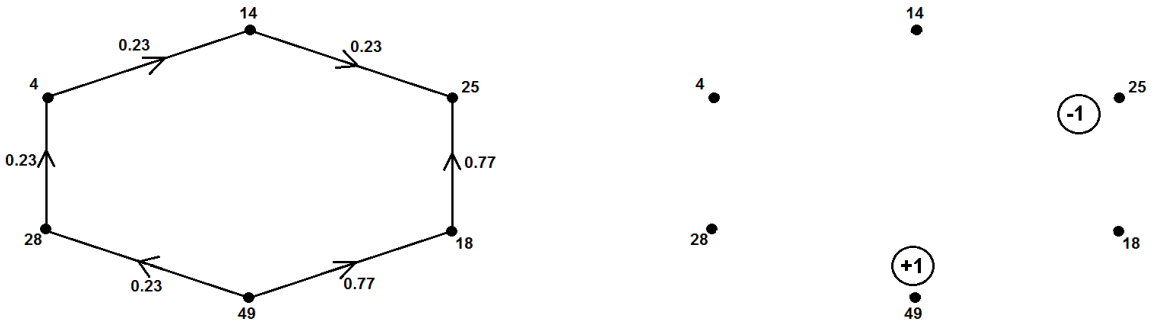

Figure 5: a) Fractional part of the LP solution. b) Flow problem.

$0 \leq y_{ij} \leq 1$), a connectivity inequality (2) associated with each connected component induced by the required arcs.

The separation algorithms that have been used to identify the connectivity, K-C, and 2-PB inequalities that are violated by the current LP solution at any iteration of the cutting plane algorithm are based on those presented in Archetti et al. [3]. Although the algorithms are not exactly the same, we omit their description for the sake of brevity. As in [3], the exact separation procedure for the connectivity inequalities is applied only if the corresponding heuristic fails to find violated inequalities. Since the separation of max-time constraints is computationally expensive, we do not look for violated inequalities of this class.

Concerning the separation of the cover inequalities, we developed a fast heuristic separation algorithm. However, we noted that the addition of these inequalities made the algorithm slower. Thus, instead of adding directly inequalities (11), we tried adding the following associated knapsack-type inequality,

$$\sum_{(i,j)\in A_P} s_{ij} y_{ij} \leq \lfloor z(LP_0) \rfloor, \tag{12}$$

and we set the "Covers" parameter in CPLEX to 1 to allow the generation of violated cover inequalities in a moderate way. Unfortunately, again the addition of these inequalities made the algorithm slower. Therefore, we decided not to use this family of inequalities.

5 Computational experiments

In this section we present the computational results obtained with the proposed branch-and-cut algorithm on a large set of instances. The procedure was coded in C/C++ using CPLEX 12.4 MIP Solver with Concert Technology 2.9 on a single thread of an Intel Core i7 at 3.4GHz with 16GB RAM. CPLEX heuristic algorithms and cut generation were turned off. The optimality gap tolerance was set to zero, and strong branching (see Applegate et al. [1]) and the best bound strategy were selected. All the experiments were carried out with a time limit of one hour.

We first performed a set of tests on the instances proposed in Archetti et al. [3] for the TOARP. They turned out to be very easy to solve when considering a single vehicle. Even the largest TOARP instances were solved in a few seconds. Therefore, we decided to generate new larger instances as follows. The vertices V are randomly chosen in a 1000×1000 square, with $|V| \in \{500, 750, 1000, 1500\}$. For each vertex we add a leaving arc to its closest

four vertices. In addition, to ensure that the graph is strongly connected, we also add three Hamiltonian tours. Each arc is declared "service arc" (either required or profitable) with probability $p_1 \in \{0.2, 0.4\}$. However, if a certain arc is labeled as service arc and the opposite arc exists, the opposite arc is automatically labeled as non-service arc. Thus, the total number of service arcs may be less than the corresponding 20% or 40%. This is done to avoid pairs of opposite arcs that are both service arcs. Among the service arcs, an arc is declared required with probability $p_2 \in \{0, 0.25, 0.50\}$. A total of five instances were generated for each combination of parameters. The generated OARP instances have between 500 and 1500 vertices, and between 3500 and 10500 arcs. The traveling time of an arc is defined as the Euclidean distance multiplied by a random number between 0.9 and 1.1. The profit of each profitable arc is chosen at random between 100 and 500. We also tried to define the profits proportional to the arc traveling times, but we did not observe any effect on the difficulty of the instances. All the instances can be found in `http://www.uv.es/corberan`, as well as the optimal value, when known, or the best feasible solution and the upper bound obtained at the root node for each instance.

In order to compute a non-trivial value of T_{max}, two DGRPs were solved for each OARP instance using the branch-and-cut algorithm described in Ávila et al. [7]. The first one considers as the set of required arcs only the required arcs of the original OARP instance, and its optimal value is denoted by z_1. In the second one, both the required and profitable arcs of the OARP instance define the set of required arcs for the DGRP, and its optimal value is denoted by z_2. The T_{max} value is defined as $z_1 + (z_2 - z_1) * r$, where r is a random number between 0.4 and 0.8. This value of T_{max} guarantees that all the required arcs, but not all the profitable ones, can be serviced.

Set	$\|A_R\|$	$\|A_P\|$	Total profit	# opt.	UB0	Opt. value	Nodes	CPU
500_2_0	0	666.8	198755.8	5/5	170018.2	170005.6	309.6	51.4
500_2_2	162.8	504.0	150307.0	5/5	132130.8	132124.4	132.6	21.1
500_2_5	321.0	345.8	102323.2	5/5	94571.0	94562.6	109.0	15.8
500_4_0	0	1198.2	362402.0	5/5	343845.6	343838.4	267.4	54.9
500_4_2	301.0	897.2	271483.6	5/5	259324.2	259315.0	340.2	64.1
500_4_5	594.6	603.6	181711.8	5/5	174110.2	174098.6	188.4	33.3

Table 2: Instances on graphs with 500 vertices and 3500 arcs

Set	$\|A_R\|$	$\|A_P\|$	Total profit	# opt.	UB0	Opt. value	Nodes	CPU
750_2_0	0	1002.4	300687.2	4/5	248495.8	247648.0	1049.0	359.9
750_2_2	245.4	757.0	226180.0	5/5	193322.2	193317.4	197.2	62.4
750_2_5	462.0	540.6	151701.8	5/5	122899.8	122890.4	364.4	95.5
750_4_0	0	1786.2	536352.0	5/5	469056.2	469051.8	476.4	217.2
750_4_2	452.0	1334.2	401209.2	5/5	310042.0	310037.8	292.2	112.0
750_4_5	695.2	910.8	273116.2	5/5	241787.4	241780.2	737.6	245.6

Table 3: Instances on graphs with 750 vertices and 5250 arcs

Tables 2 to 5 show the computational results obtained with the branch-and-cut algorithm on the instances with 500, 750, 1000, and 1500 vertices, respectively. In each table, the first

Set	$\|A_R\|$	$\|A_P\|$	Total profit	# opt.	UB0	Opt. value	Nodes	CPU
1000_2_0	0	1331.8	401591.4	5/5	356423.0	356414.8	1718.2	899.2
1000_2_2	337.4	994.4	298978.2	5/5	255688.2	255683.6	518.0	250.6
1000_2_5	654.8	677.0	204564.8	5/5	175125.6	175099.0	813.8	335.4
1000_4_0	0	2373.2	709900.8	5/5	642066.8	642064.6	146.6	189.1
1000_4_2	581.8	1791.4	533480.2	5/5	433198.4	433194.6	386.0	266.8
1000_4_5	1182.6	1190.6	355550.2	5/5	283376.2	283371.2	452.4	286.4

Table 4: Instances on graphs with 1000 vertices and 7000 arcs

Set	$\|A_R\|$	$\|A_P\|$	Total profit	# opt.	UB0	Opt. value	Nodes	CPU
1500_2_0	0	2023.2	606497.8	4/5	506001.4	505955.6	740.3	1158.0
1500_2_2	502.0	1521.2	457016.4	5/5	368421.4	368417.2	473.2	541.3
1500_2_5	1011.8	1011.4	301687.8	5/5	246308.6	246291.4	201.0	226.7
1500_4_0	0	3568.4	1072711.4	4/5	964030.3	964023.4	390.5	917.9
1500_4_2	876.6	2691.8	810600.6	5/5	676398.8	676395.4	595.8	923.5
1500_4_5	1793.2	1775.2	533358.6	5/5	465266.8	465263.8	615.6	821.9

Table 5: Instances on graphs with 1500 vertices and 10500 arcs

column shows the name of the set of instances, where the first number refers to the number of vertices, the second one to the value of p_1, and the third one to p_2. The number of instances in each set is equal to 5. Columns 2 and 3 give the average number of required and profitable arcs of each set, respectively. The average sum of the profits of all the profitable arcs is given in column 'Total profit'. Column labeled '# opt.' shows the number of instances that were solved to optimality. Columns 'UB0' and 'Opt. value' report the average upper bound obtained at the end of the root node and the average optimal value (or best solution found), respectively. Finally, columns 'Nodes' and 'CPU' give the average number of nodes of the branch-and-cut tree and the computing times (in seconds) for the instances that were solved optimally.

Note that the difference between the total profit and the optimal value shows that the instances are not trivial, since there is an important gap between the two.

In all the groups of instances, the most difficult ones are those with $p_1 = 0.2$ and $p_2 = 0$, that is, approximately 20% of the arcs are service arcs and all of them are profitable arcs. We think this is due to the difficulty of making the solution connected when there are very few service arcs and all these arcs are profitable, which may or may not be serviced. For the remaining instances, it seems that those with $p_1 = 0.4$ are more difficult than those with $p_1 = 0.2$.

We conclude that the branch-and-cut algorithm performs very well, since it is capable of solving all the instances, but three, to optimality in short computing times. Moreover, the unsolved instances "750_2_0_4", "1500_2_0_1", and "1500_4_0_4" presented a final gap of 2.08%, 0.0009%, and 0.0026%, respectively, within the time limit of one hour. Note also that the gap between the upper bound at the end of the root node and the optimal value is very small, which indicates that the polyhedral description is tight.

6 Conclusions

In this paper we presented the Orienteering Arc Routing Problem, which is an arc routing problem with profits where a single route has to be built with the aim of maximizing the total collected profit while satisfying a maximum time duration of the route. An integer programming formulation and several families of valid inequalities have been proposed. A branch-and-cut algorithm, including a heuristic based on the information provided by the fractional LP solutions, has been implemented. The computational results show that the behavior of the algorithm is excellent since it is able to solve instances with up to 1500 vertices and 10500 arcs in a few minutes of computing time. We recall that in the TOARP with four vehicles only instances with up to 100 vertices and 800 arcs could be optimally solved. This leads us to think that the OARP can be considered a "well-solved" problem.

Acknowledgments: Ángel Corberán, Isaac Plana and José M. Sanchis wish to thank the Ministerio de Economía y Competitividad of Spain (MTM2012-36163-C06-02) and the Generalitat Valenciana (project GVPROMETEO2013-049) for its support.

References

[1] D. Applegate, R.E. Bixby, V. Chvátal and W. Cook (1995), Finding Cuts in the TSP. Technical report, DIMACS 95–05.

[2] C. Archetti, Á. Corberán, I. Plana, J.M. Sanchis and M.G. Speranza (2013), A Matheuristic for the Team Orienteering Arc Routing Problem. Working paper WPDEM 2013/9, Dipartimento di Economia e Management, Università degli Studi di Brescia.

[3] C. Archetti, Á. Corberán, I. Plana, J.M. Sanchis and M.G. Speranza (2014), The Team Orienteering Arc Routing Problem. *Transportation Science* 48, 442-457.

[4] C. Archetti and M.G. Speranza (2014), Arc Routing Problems with Profits. Accepted for publication as Chapter 12 in Á. Corberán and G. Laporte (eds.): Arc Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, SIAM, Philadelphia (forthcoming).

[5] C. Archetti and M.G. Speranza (2014), A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization* 2, 223-246.

[6] C. Archetti, M.G. Speranza and D. Vigo (2014), Vehicle Routing Problems with Profits. Accepted for publication as Chapter 12 in P. Toth and D. Vigo (eds.): Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOS-SIAM Series on Optimization 18, SIAM, Philadelphia (forthcoming).

[7] T. Ávila, Á. Corberán, I. Plana and J.M. Sanchis (2014), The Stacker Crane Problem and the Directed General Routing Problem. *Networks*. To appear.

[8] D. Feillet, P. Dejax and M. Gendreau (2005), Traveling Salesman Problems with Profits. *Transportation Science* 39, 188-205.

[9] A.N. Letchford (1997), New Inequalities for the General Routing Problem. *European Journal of Operational Research* 96, 317-322.

[10] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe and D. Van Oudheusden (2011), The planning of cycle trips in the province of East Flanders. *OMEGA* 39, 209-213.