C. Archetti, Á. Corberán, I. Plana, J.M. Sanchis, M. G. Speranza

A Matheuristic for the Team Orienteering Arc Routing Problem

WPDEM 2013/9

Via S. Faustino 74/b, 25122 Brescia – Italy
Tel 00390302988742 Fax 00390302899703
email: paqrircadem@eco.unibs.it
A Matheuristic for the Team Orienteering Arc Routing Problem

Claudia Archetti\(^{(1)}\) Ángel Corberán\(^{(2)}\) Isaac Plana \(^{(3)}\)
José M. Sanchís \(^{(4)}\) M. Grazia Speranza\(^{(1)}\)

\(^{(1)}\) Department of Economics and Management, University of Brescia, Italy, archetti@eco.unibs.it, speranza@eco.unibs.it
\(^{(2)}\) Department of Statistics and Operations Research, University of Valencia, Spain, angel.corberan@uv.es
\(^{(3)}\) Department of Mathematics for Economics and Business, University of Valencia, Spain, isaac.plana@uv.es
\(^{(4)}\) Department of Applied Mathematics, Polytechnical University of Valencia, Spain, jmsanchis@mat.upv.es

Abstract

In the Team Orienteering Arc Routing Problem (TOARP) the potential customers are located on the arcs of a graph and are to be chosen on the basis of an associated profit. A limited fleet of vehicles is available to serve the chosen customers. Each vehicle has to satisfy a maximum route duration constraint. The goal is to maximize the profit of the served customers. We propose a matheuristic for the TOARP and test it on a set of benchmark instances for which the optimal solution or an upper bound are known. The matheuristic finds the optimal solutions on all, except one, instances of one of the three classes of tested instances (with up to 27 vertices and 296 arcs). The average error on all instances for which the optimal solution is available is 0.64%.

Keywords: Team Orienteering Problem, Arc Routing Problem, Matheuristic.

1 Introduction

In arc routing problems, customers are located on arcs and the routes of minimum cost have to be identified. The basic problems of this class are the Chinese Postman Problem, where all edges or arcs have to be visited, and the Rural Postman Problem, where only some edges or arcs are required to be visited. There are several applications of arc routing problems where the edges or arcs to be visited are not given and have instead to be selected on the basis of a profit. In fact, for any arc routing problem with given customers to visit (see the book edited by Dror [16]) a version where customers have to be chosen is likely to have interesting applications. Typical applications of arc routing problems include road maintenance, garbage collection, and mail delivery. In all these cases, if it is not possible to visit all customers in a day because vehicles, people
or time are not sufficient, one has to choose the most valuable customers to serve.

In the routing literature where customers are located on vertices, the problems where the customers have to be selected on the basis of their profit are called *routing problems with profits* (see [17]). We call the arc routing counterpart of this class *arc routing problems with profits*. A well studied problem in the former class is the Team Orienteering Problem (TOP), where a fleet of uncapacitated vehicles is available, with a time duration constraint on each route, and the problem is to select a set of customers to maximize the total profit of the customers served. In this paper we study the arc routing counterpart of the TOP, the Team Orienteering Arc Routing Problem (TOARP), that was introduced in [5]. In the TOARP, customers with an associated profit are located on arcs and a fleet of vehicles with a time duration constraint on each route are given. The problem consists in choosing the customers and in designing the routes in such a way that the collected profit is maximized. An interesting application of the TOARP is in truck-load transportation, where some of the customers are to be served whereas others may be postponed or not served at all. For example the service of the least profitable customers may be outsourced. In [5] an extended polyhedral study is presented. The proposed branch-and-cut algorithm solves instances with up to 100 vertices, 800 arcs, and 4 vehicles to optimality using the values provided by the heuristic described in this paper.

In this paper we address the heuristic solution of the TOARP by means of an algorithm that combines a tabu search scheme, to escape from local optima, and the optimal solution of integer programming models to intensify the search in some areas of the solution space. The combination of a heuristic or metaheuristic scheme with integer linear programming models has been recently explored by several authors. A survey is presented by Ball [9]. These heuristics are named in different ways. In the survey [9] they are simply called *heuristics based on mathematical programming*, in other cases (see [26]) they go under the generic name of *hybrid heuristics*. The name *matheuristics* has been created ad hoc for this class of heuristics (see [20]) and this is the name we will use in this paper. The matheuristic was tested on the instances introduced in [5].

The paper is organized as follows. In Section 2 we introduce the TOARP whereas in Section 3 we describe the general scheme of the matheuristic and its components. The computational results are presented and discussed in Section 4.
2 The team orienteering arc routing problem

The TOARP is defined on a directed graph. In general, the graph is not complete. A numerical value representing the traversal time is associated with each arc. Only some of the arcs represent customers. Some customers have to served, and are called required whereas some others may be served if beneficial. A profit is associated with each customer of the latter set. We call these customers profitable. A limited fleet of vehicles is available. Each vehicle starts its route at the depot, traverses a set of arcs and ends its route at the depot. Each route cannot exceed a maximum time duration. The goal is to choose a set of the profitable customers and to design the routes of the vehicles such that the required and chosen profitable customers are served, the time duration constraints of the routes are satisfied and the total profit collected is maximized.

More formally, a directed graph $G = (V, A)$ is given, where $V = \{1, \ldots, n\}$ is the set of vertices and $A$ is the set of arcs. Vertex 1 is the depot, that is the starting and ending vertex of each route. A traveling time $c_{ij}$ is associated with each arc $(i, j) \in A$. Some arcs represent customers. The set $A_R \subseteq A$ represents customers that have to be serviced, whereas $A_P \subseteq A$ represents the set of profitable customers. A nonnegative profit $s_{ij}$ is associated with each arc $(i, j) \in A_P$. A fleet of $K$ vehicles is available. The route of each vehicle cannot exceed a maximum time duration $T_{\text{max}}$. The profit of any profitable customer can be collected by one vehicle at most. The objective of the TOARP is to maximize the total profit collected.

3 A matheuristic for the TOARP

In this section we present a matheuristic for the solution of the TOARP that we call MAT (MAtheuristic for TOARP).

In the following, we say that a profitable arc is served by a vehicle if the vehicle traverses the arc and collects the corresponding profit. In MAT, there is no distinction between required and profitable arcs. A very large profit is assigned to the arcs in $A_R$ and all arcs in $A_P \cup A_R$ are considered, and called, profitable. Thus $A_P := A_P \cup A_R$. Obviously, this means that the solution given by MAT may be infeasible as one or more arcs in $A_R$ may be not served. However, this is unlikely to happen given that each arc in $A_R$ is associated with a very large profit.

The structure of MAT resembles a Variable Neighborhood Search (VNS). The main idea of a VNS (see [22]) is to use different neighborhoods during the search. Given a solution $s$, a neighbor solution $s'$ is generated according to these neighborhoods, and a local search is then applied to $s'$ to obtain a
possibly better solution \( s'' \). If \( s'' \) is better than \( s \), then \( s'' \) becomes the new incumbent; otherwise, a different neighborhood is considered in order to try to improve upon solution \( s \). Let \( N^{(k)}, k = 0, \ldots, k_{\text{max}} \), denote a set of neighborhoods, where \( N^{(k)}(s) \) is the set of solutions in the \( k \)-th neighborhood of \( s \). A basic VNS is described in Figure 1.

Choose an initial solution \( s \);
set \( k = k_{\text{max}} \)
Repeat the following steps until a stopping criterion is met

- **shaking**: Generate \( s' \) at random in \( N^{(k)}(s) \)
- **local search**: Apply a local search on \( s' \) using \( N^{(0)} \). Let \( s'' \) be the resulting solution
- **update**: If \( s'' \) is better than \( s \) then set \( s = s'' \) and \( k = k_{\text{max}} \), else set \( k = k - 1 \).
  
  If \( k = 0 \) set \( k = k_{\text{max}} \).

Figure 1: Basic VNS

Neighborhood \( N^{(0)} \) is used in the local search phase, but not in the shaking phase. Solutions in \( N^{(0)}(s) \) have a structure that is usually much closer to \( s \) than that of the solutions in \( N^{(k)}(s) \) with \( k > 0 \). For this reason, a move from \( s \) to a solution \( s' \in N^{(k)}(s) \ (k > 0) \) is often called a **jump**.

MAT is a combination of a VNS with the optimal solution of different Integer Linear Programming (ILP) models. The scheme of the VNS is an adaptation to the TOARP of the VNS proposed in Archetti et al. [6] to solve the Undirected Capacitated Arc Routing Problem with Profits (UCARPP).

The local search phase of the VNS consists of a short tabu search, that we sketch later.

A general scheme of MAT is given in Figure 2.

INITIAL SOLUTION \( (s) \)

**While** a stopping criterion is not met **do**

**JUMP** \( (k,s) \)

**INTERNAL TABU SEARCH** \( (s) \)

**If** \( s \) is better than \( s_{\text{best}} \) **then**
FIRST IMPROVEMENT $(s)$. $s_{best} \leftarrow s$.

End If

SECOND IMPROVEMENT $(s)$. $s_{best} \leftarrow s$.

End While

Figure 2: MAT: The matheuristic for the TOARP

The FIRST IMPROVEMENT $(s)$ and SECOND IMPROVEMENT $(s)$ procedures represent innovative features of MAT with respect to the VNS scheme proposed in [6] for the UCARPP. The purpose of both procedures is to improve the best solutions obtained through the internal tabu search. Their introduction has remarkably improved the performance of the algorithm.

Before describing the different components of MAT, we introduce some notations and definitions.

The profit $S(C)$ of a set of profitable arcs $C \subseteq A_P$ is the total profit $\sum_{a \in C} s_a$. We denote by $L_r$ the set of profitable arcs served by route $r$ and with $A_r$ the set of all arcs traversed by route $r$. The profit $S(r)$ of a route $r$ is defined as the total profit of the profitable arcs served by the route, i.e., $S(r) = S(L_r)$. The duration $T(r) = \sum_{a \in A_r} c_a$ of a route $r$ is its total travel time. A route $r$ is feasible if it starts and ends at the depot and $T(r) \leq T_{max}$.

For a set $R$ of routes, $S(R) = \sum_{r \in R} S(r)$ is the total profit of the routes in $R$ and $L(R) = \bigcup_{r \in R} L_r$ is the set of profitable arcs served by the routes in $R$.

A solution $s$ is a set of routes such that each profitable arc is served by exactly one route. A solution $s$ is feasible if each route in $s$ is feasible. Although in a solution $s$ any arc (profitable or not) may be traversed more than once, the profit of a profitable arc is collected exactly once. We denote by $R_P(s)$ the set of the $K$ most profitable routes in $s$ (or the set of all routes in $s$ if they are less than or equal to $K$), and $R_N(s)$ the set of all remaining routes. The aim of the TOARP is to determine a feasible solution $s$ that maximizes $S(R_P(s))$.

The profitable arcs in $L(R_N(s))$ do not belong to the $K$ most profitable routes in $s$, but are already organized into routes. The reason why we keep these arcs organized into routes is that it is much easier to have a new route with a high profit by inserting profitable arcs in one of the routes in $R_N(s)$ than to create a new route from scratch. Although this requires an additional effort with respect to keeping only the $K$ most profitable routes, it turned out to be beneficial to the efficiency of the heuristic.

In a solution $s$, we denote by $r_a(s)$ the route serving the profitable arc $a$. For a profitable arc $a$ and a route $r \neq r_a(s)$, we denote by $r + a$ the route obtained by adding $a$ to $r$. Similarly, given a route $r$ and a profitable
arc $a$ served by $r$, we denote $r - a$ the route obtained from $r$ by removing $a$. Before describing the main structure of MAT, we explain how the insertion and removal of profitable arcs from routes is performed.

The evaluation of the insertion of an arc $a$ into a route $r$ is performed by calling the ADD operator by Hertz, Laporte, Mittaz (2000) for the Undirected Capacitated Arc Routing Problem (UCARP), adapted to the case of a directed graph. This operator makes use of the SHORTEN operator that tries to reduce the travel time of the route. Let $T(r)$ be the travel time of the route before inserting $a$ and $T(r + a)$ the travel time of the route once $a$ has been inserted into $r$ using the ADD operator. The insertion cost is $\Gamma = T(r + a) - T(r)$. Obviously, if $a \in A_r$, then the insertion cost is $\Gamma = 0$.

The evaluation of the removal of arc $a$ from route $r$ is performed by calling the DROP operator by Hertz, Laporte, Mittaz (2000) for the UCARP, adapted to the case of a directed graph. Again this operator makes use of the SHORTEN operator. Indicating by $T(r - a)$ the travel time of the route once $a$ has been removed from $r$ using the DROP operator, the removal saving is $\Delta = T(r) - T(r - a)$.

We now describe in details the procedures that compose MAT.

### 3.1 Initial solution ($s$)

The initial solution is constructed by means of a greedy procedure that ranks the arcs in order of non-decreasing profit. The greedy algorithm then builds the routes sequentially adding arcs to each route until it is feasible and creating a new route when the time constraint is violated. When all the profitable arcs have been inserted into a route, the $K$ most profitable routes compose the initial set $R_P(s)$ and the remaining routes the initial set $R_N(s)$.

### 3.2 Internal tabu search ($s$)

Once the initial solution has been obtained, a tabu search is executed that uses the following three moves:

- **1-move**: In a 1-move, a profitable arc $a$ is moved from its route $r_a(s)$ to a route $r \neq r_a(s)$. Route $r$ may be an empty route. Hence, $r_a(s)$ and $r$ are replaced by $r_a(s) - a$ and $r + a$, respectively.

- **swap-move**: Let $a$ and $a'$ be two profitable arcs on two different routes. A swap-move consists in replacing $r_a(s)$ and $r_{a'}(s)$ by $(r_a(s) - a) + a'$ and $(r_{a'}(s) - a') + a$, respectively.
• sequence move : Let \( I \) be a sequence of profitable arcs served consecutively in a route \( r \in R_N(s) \). Choose a route \( r' \in R_p(s) \). For each \( a \in I \), remove \( a \) from \( r \) and insert it in \( r' \). If \( T(r') > T_{\text{max}} \), then choose a sequence \( I' \) of profitable arcs served in \( r' \) such that \( S(I') < S(I) \). For each \( a \in I' \), remove \( a \) from \( r' \) and insert it in a route \( r'' \in R_N(s) \).

Note that the sequence move may be seen as a sort of multiple 1-moves or swap-moves. It is performed only if it improves the current solution \( (S(I) > S(I')) \) and only every 5 iterations. The sequence move was not implemented in the VNS proposed in Archetti et al. [6] to solve the UCARPP. We introduced this move as it is able to remarkably improve the results obtained with the first two moves only.

A temporary tabu status forbids profitable arcs to be removed from routes in which they have been previously inserted. In particular, when arc \( a \) is inserted in route \( r \), it is tabu to remove it for a number of iterations equal to:

\[
l = \sqrt{\lambda \sigma / 4} + |A_p| \star \sqrt{K / 8},
\]

where \( \lambda = |V| \star |R| \), \( R \) is the set of routes in the initial solution and \( \sigma \) is a random number in \([0, 1]\). A similar formula has been used in [6] in the algorithms proposed for the UCARPP. It links the number of tabu iterations to the number of vertices and profitable arcs and to the number of routes. Parameter \( \sigma \) is introduced in order to have a certain variability in the number of tabu iterations.

The quality of feasible solutions is measured by combining the four following functions:

• \( f_1(s) \) : the total profit \( S(R_p(s)) \) of the routes in \( R_p(s) \);

• \( f_2(s) \) : the total duration \( \sum_{r \in R_p(s)} T(r) \) of the routes in \( R_p(s) \);

• \( f_3(s) \) : the number of non empty routes in \( s \);

• \( f_4(s) \) : the total duration \( \sum_{r \in R_N(s)} T(r) \) of the routes in \( R_N(s) \).

A feasible solution \( s \) is considered as better than a feasible solution \( s' \) if \( f_1(s) > f_1(s') \), or \( f_1(s) = f_1(s') \) and \( f_2(s) < f_2(s') \), or \( f_1(s) = f_1(s') \), \( f_2(s) = f_2(s') \) and \( f_3(s) < f_3(s') \), or \( f_1(s) = f_1(s') \), \( f_2(s) = f_2(s') \), \( f_3(s) = f_3(s') \) and \( f_4(s) < f_4(s') \).
3.3 First improvement $(s)$

Every time a new best solution $s$ is found, we intensify the search in the neighborhood of this solution with the aim to improve it further. To do that, we try to reduce the time duration of the routes of $s$ to increase the probability of inserting additional profitable arcs and then apply the sequence move. We have chosen the sequence move for this intensification phase as it proved to be effective in finding high quality solutions which are ‘close enough’ to the starting solution $s$ but may be missed by the simpler 1-moves and swap-moves. To reduce the time duration of the routes of $s$ for each route $r$ in $R_p(s)$ we solve an Asymmetric Traveling Salesman Problem (ATSP) on all the profitable arcs currently served by the route. Each vertex of the graph over which the ATSP is solved corresponds to a profitable arc served by route $r$. The depot is also a vertex of the graph. The cost of arc $(i, j)$ is the cost of the shortest path between the head of the profitable arc represented by vertex $i$ and the tail of the profitable arc represented by vertex $j$. The ATSP is solved to optimality using the Miller-Tucker-Zemlin formulation [21] which has a polynomial number of constraints. This way, the length of each route is reduced to the minimum and we increase the probability of inserting profitable arcs in the most profitable routes. After that, the sequence move is called iteratively until it improves the current solution.

3.4 Second improvement $(s)$

Once the Internal Tabu Search $(s)$ procedure is terminated, a new jump is made. The aim of the jump is to completely destroy the current solution and build a new and completely different solution on which the Internal Tabu Search $(s)$ is started again. A detailed description of the Jump $(k,s)$ procedure is made in Section 3.5. The aim of the Second Improvement $(s)$ procedure is to intensively exploit the possibility of improving the current best solution before moving to a different part of the solution space with a jump. This procedure consists in the solution of two ILP models. The reason why we opted for the optimal solution of ILP models instead of applying more traditional simple operators is that this strategy has been computationally proved to be extremely effective in improving solutions especially when the problem structure is complex (like in the TOARP) while a simple operator or even a combination of simple operators do not have the same degree of effectiveness. This strategy has already been proved effective in the solution of other complex routing problems (see [4], [8] and [7]).

The two ILP models deeply differ in terms of their aim. The first ILP model (ILP1) tries to select the $K$
most profitable routes among the \( R \) ones generated so far. Thus, the solution space is defined by the entire set \( R \) of routes generated from the beginning of the algorithm (which thus may be quite large). The ILP1 is a set covering problem where the \( K \) best routes are chosen. The second model, ILP2, focuses on the best solution \( s \) found so far and tries to increase the profit collected by the routes in \( R_P(s) \) by inserting profitable arcs or sequences of profitable arcs and removing less profitable arcs. ILP2 is run only if ILP1 is able to improve the current best solution. In fact, if this is not the case, the current best solution has already been processed by the FIRST IMPROVEMENT \( (s) \) procedure and thus the probability of improving this solution further through ILP2 is low. If instead a new best solution is found by ILP1, then ILP2 makes a deep search around this solution to try to increase the corresponding profit.

Let us now describe the two models in more detail. We define \( R \) as the set of all routes generated by the VNS until the current iteration and let \( b_{ar} = 1 \) if the profitable arc \( a \) is traversed by route \( r \in R \). ILP1 makes use of binary variables \( z_a = 1 \) if profitable arc \( a \) is served, 0 otherwise, and variables \( x_r = 1 \) if route \( r \in R \) is used. ILP1 is formulated as follows:

\[
\text{Maximize} \quad \sum_{a \in A_P} s_a z_a \\
\text{s.t.:} \\
\sum_{r \in R} b_{ar} x_r \geq z_a \quad a \in A_P \\
\sum_{r \in R} x_r \leq K \\
x_r \in \{0; 1\} \quad r \in R \\
z_a \in \{0; 1\} \quad a \in A_P
\]

If ILP1 is able to find a new best solution, then a ILP2 is applied on the new solution \( s \) in order to try to further improve it. ILP2 tries to insert in \( R_P(s) \) profitable arcs or pairs of profitable arcs. A pair of profitable arcs is considered for insertion only if the distance between the two arcs is lower than a given threshold \( T \) set to

\[
T = \frac{T_{\text{max}} K}{|L(R_N(s))|}.
\]

The distance between arcs \( a \) and \( a' \) is set as the minimum between the cost of the shortest path between
the head of $a$ and the tail of $a'$ and the cost of the shortest path between the head of $a'$ and the tail of $a$.

Let $\mathcal{S}$ be the set of all sets composed by single profitable arcs and pairs of profitable arcs with a distance lower than the given threshold served in $R_N(s)$. Note that each set $i \in \mathcal{S}$ may correspond to a single profitable arc or to a pair of profitable arcs. We denote by $S(i)$ the profit of set $i \in \mathcal{S}$. Also, let $\Gamma_{ir}$ be the cost of inserting set $i \in \mathcal{S}$ in route $r \in R_P(s)$ and $\Delta_{ar}$ be the removal saving gained if arc $a$ is removed from route $r \in R_P(s)$. Finally, let $b_{ar}$ be equal to 1 if profitable arc $a$ is currently served by route $r \in R_P(s)$, $g_{ai}$ be equal to 1 if $a \in A_P$ is in set $i \in \mathcal{S}$ and $f_{air}$ be equal to 1 if the profitable arc $a$ is traversed by $r \in R_P(s)$ when set $i \in \mathcal{S}$ is inserted in $r$ ($a$ is traversed along the path that links route $r$ with the arcs in set $i$). ILP2 makes use of the following binary variables: $v_{ir}$ which takes value 1 if set $i \in \mathcal{S}$ is inserted in $r \in R_P(s)$, $w_{ar}$ which takes value 1 if arc $a$, currently served by route $r \in R_P(s)$, is removed from $r$, and $z_{air}$ which takes value 1 if arc $a$ is served by route $r \in R_P(s)$ when set $i \in \mathcal{S}$ is inserted in $r$. ILP2 is formulated as follows:

$$\text{Maximize} \quad \sum_{i \in \mathcal{S}} \sum_{r \in R_P(s)} \left( S(i)v_{ir} + \sum_{a \in \mathcal{A}} w_{ar}z_{air} - w_{ar} \right)$$

s.t.:

$$\sum_{i \in \mathcal{S}} \Gamma_{ir}v_{ir} - \sum_{a \in A_P} \Delta_{ar}w_{ar} \leq T_{\text{max}} - T(r) \quad r \in R_P(s) \quad (5)$$

$$v_{ir} \leq 1 - g_{ai}b_{ar} \quad a \in A_P, i \in \mathcal{S}, r \in R_P(s) \quad (6)$$

$$w_{ar} \leq b_{ar} \quad a \in A_P, r \in R_P(s) \quad (7)$$

$$z_{air} \leq f_{air}v_{ir} \quad a \in A_P, i \in \mathcal{S}, r \in R_P(s) \quad (8)$$

$$\sum_{i \in \mathcal{S}} (b_{ar} - w_{ar} + \sum_{i \in \mathcal{S}} (g_{ai}v_{ir} + f_{air}z_{air})) \leq 1 \quad a \in A_P \quad (9)$$

$$\sum_{a \in A_P} w_{ar} + \sum_{i \in \mathcal{S}} v_{ir} \leq \Theta \quad r \in R_P(s) \quad (10)$$

$$v_{ir} \in \{0,1\} \quad i \in \mathcal{S}, r \in R_P(s) \quad (11)$$

$$w_{ar} \in \{0,1\} \quad a \in A_P, r \in R_P(s) \quad (12)$$

$$z_{air} \in \{0,1\} \quad a \in A_P, i \in \mathcal{S}, r \in R_P(s) \quad (13)$$

where $\Theta$ is a parameter set to 5 in our tests.

The objective function aims at maximizing the profit of the routes in $R_P(s)$. Constraints (5) aim at ensuring feasibility of the new solution. However, the left-hand side of inequality (5), with the addition of $T(r)$, does not give the exact value of the cost of the new route. Thus, the new route may be infeasible even if it satisfies constraint (5). Constraints (6) establish that a sequence of arcs can be inserted into a route only if all arcs in the sequence are not served by the route while from (7) an arc $a$ can be removed from route $r$ only if $r$ serves $a$. Inequalities (8) impose that variable $z_{air}$ is set to 1 only if sequence $i$ is inserted in route
(9) guarantees that each profitable arc is served at most once. Finally, with (10) we impose that a limited number of arcs can be removed or inserted from a route. This is done in order to avoid a poor estimation of the new cost of the route in constraint (5).

A note has to be made on constraint (5). The left-hand side of (5) is an approximation of the new cost of the route and not the exact cost for the following reason. When we insert and/or remove more than one arc in the same route, then the new cost of the route may be different from the one obtained by simply summing up the corresponding insertion costs or removal savings. For example, suppose that we remove two profitable arcs $a$ and $a'$ from the same route $r$ and these arcs are visited consecutively in route $r$ (meaning that no other profitable arc is served between the two), with $a$ visited before $a'$. Then the cost of the arcs connecting the head of $a$ with the tail of the $a'$ are accounted for both in $s_a$ and in $s_{a'}$, and thus twice in the left-hand side of (5).

### 3.5 Jump ($k, s$)

The internal tabu search phase is stopped when 400 iterations without improvement of the best solution found are elapsed. Then, in order to explore various regions of the solution space, a jump is made. We use two kinds of jumps. One consists in performing a sequence of $k$ 1-moves from $R_N(s)$ to $R_P(s)$, while the other moves a set $U$, with $|U| = k$, of profitable arcs from $R_P(s)$ to $R_N(s)$, and a set $W$ of profitable arcs from $R_N(s)$ to $R_P(s)$, such that $S(U) \leq S(W)$. Note that, the larger the value of $k$, the more different the new solution is with respect to the previous one.

Typically, the solution obtained after a jump is infeasible. In order to recover feasibility we first optimize the length of each route by solving an ATSP and then apply 1-moves and swap-moves between the routes in $R_P(s)$ in order to reduce the infeasibility. Only moves that reduce infeasibility are implemented. If some routes are still infeasible after all such moves, then we call the REMOVE INFEASIBILITY procedure which is the adaptation to the directed case of the procedure proposed in [6] for the UCARPP. The procedure randomly removes arcs from infeasible routes until each route becomes feasible. The removed arcs are inserted in existing routes (if this is feasible) or in new routes.

At the end of the procedure that recovers the feasibility we may obtain a solution of very poor quality which may constitute a bad starting point for the following tabu search phase. In order to improve this starting solution, a further ILP model (ILP3) is solved on the solution generated after the jump. ILP3 is a simplification of ILP2. In fact, applying ILP2 at each jump proved to be too time consuming. As ILP2,
ILP3 tries to increase the profit collected by the routes in \( R_p(s) \) by inserting profitable arcs and removing less profitable arcs. However, contrary to what happens in ILP2, in ILP3 it is possible to only insert or remove single profitable arcs from the routes in \( R_p(s) \). With a similar notation as in ILP2, let \( b_{ar} \) be equal to 1 if profitable arc \( a \) is currently served by route \( r \in R_p(s) \), \( \Gamma_{ar} \) be the cost of inserting arc \( a \in A_P \) in route \( r \in R_p(s) \) and \( \Delta_{ar} \) be the removal saving gained if arc \( a \) is removed from route \( r \in R_p(s) \). ILP3 makes use of the following binary variables: \( v_{ar} \) which takes value 1 if arc \( a \in A_P \) is inserted in \( r \in R_p(s) \) and \( w_{ar} \) which takes value 1 if arc \( a \), currently served by route \( r \in R_p(s) \), is removed from \( r \). ILP3 is formulated as follows:

Maximize \( \sum_{a \in A_P} \sum_{r \in R_p(s)} (s_a v_{ar} - s_a w_{ar}) \)

s.t.:

\[
\sum_{a \in A_P} (\Gamma_{ar} v_{ar} - \Delta_{ar} w_{ar}) \leq T_{\text{max}} - T(r) \quad r \in R_p(s) \tag{14}
\]

\[
v_{ar} \leq 1 - b_{ar} \quad \forall a \in A_P, \ r \in R_p(s) \tag{15}
\]

\[
w_{ar} \leq b_{ar} \quad \forall a \in A_P, \ r \in R_p(s) \tag{16}
\]

\[
\sum_{r \in R_p(s)} (b_{ar} - w_{ar} + v_{ar}) \leq 1 \quad a \in A_P \tag{17}
\]

\[
\sum_{a \in A_P} w_{ar} + \sum_{i \in S} v_{ir} \leq \Theta \quad r \in R_p(s) \tag{18}
\]

\[
v_{ir} \in \{0;1\} \quad i \in \mathcal{S}, r \in R_p(s) \tag{19}
\]

\[
w_{ar} \in \{0;1\} \quad a \in A_P, r \in R_p(s) \tag{20}
\]

where \( \Theta \) is a parameter that takes value 3 in our tests.

Note that, as the solution obtained from the solution of ILP2 and ILP3 may be infeasible, the REMOVE INFEASIBILITY procedure is called in order to recover feasibility.

Finally, note that, even if ILP3 is applied, the solution obtained after a jump may be extremely poor and starting a tabu search phase from a very poor solution may be a waste of time. Thus, the tabu search phase is started only if the profit associated with the solution obtained after the jump is at least equal to \( z_{\text{best}} \beta \), where \( z_{\text{best}} \) is the value of the best solution found so far. \( \beta \) is initialized to 0.9. After 5 consecutive trials for which the value of the solution is lower than \( z_{\text{best}} \beta \), the value of \( \beta \) is updated to \( \beta \leftarrow 0.9 \beta \).
4 Computational results

MAT was tested on the set of instances proposed in [5]. These instances are generated from benchmark RPP instances proposed by Hertz et al. [23]. In these instances, vertices are points in the Euclidean plane and edge costs are defined by the Euclidean distances. The set of edges is defined according to three different methods. In the first method (R class instances) the edges are generated randomly in the plane. The second method (G class instances) generates a set of edges with cost 1, defining a uniform grid. The third method (D class instances) chooses edges defining a graph where all the vertices have degree 4. These instances have been transformed into TOARP instances as follows. For each edge $\{i, j\}$, two arcs $(i, j)$ and $(j, i)$ with the same cost are generated. If the edge was not required, both arcs are not required. If it was required, one direction is randomly selected and the corresponding arc is required (with probability $p$) or profitable (with probability $1 - p$), while the opposite one is not required. The profit associated with the traversal of a profitable arc is defined as its cost, because it seems reasonable that the profit obtained by serving an arc is proportional to the traversal cost.

The code was written in Visual C++ 2010 and tests were run on an AMD Athlon (tm) 64 X2 Dual Core Processor 5600+ 2.89 GHz, 3.37 GB RAM. The stopping criterion was set to 30 minutes of computing time. We recall in Table I the instance characteristics as reported in [5]. To calculate optimal solution of the ATSP, ILP1, ILP2 and ILP3 we used CPLEX 12.2.0. The maximum time allowed for the solution of each ATSP was set to 30 seconds while 30 minutes are allowed for the solution of ILP1, ILP2 and ILP3.

For 204, out of 207, instances with 2 vehicles the optimal solution is known, while the optimal solution is known for 188 and 157, out of 207, instances for 3 and 4 vehicles, respectively.

The following tables summarize the results. Table II refers to the instances of class R, Table III to class D and Table IV to class G. In each table, the instances are classified by the number of vehicles $K$ (on the rows) and by parameter $p$ (on the columns). For each class of instances we report: the number of instances solved to optimality by the branch-and-cut algorithm proposed in [5] (# solved), the number of optimal solutions found by MAT (# opt), the average percentage gap of the solution found by MAT with respect to the upper bound found by the branch-and-cut algorithm (Av. gap) and the average percentage gap with respect to the optimal solution, when found by the branch-and-cut algorithm (Av. gap*).

Looking at Tables II - IV it is possible to note that the easiest class of instances, both for the heuristic and the exact algorithm, is class R where all instances are solved to optimality by the branch-and-cut algorithm and MAT always finds the optimal solution except for one instance with $K = 2$ and $p = 0.5$. For this instance,
the error with respect to the optimal solution is 28.95% (which gives an average error over five instances of 5.79% in the last column of Table II). Such a large error is due to the fact that the solution space is narrow and either the optimal solution is found or the gap for any non-optimal solution is large. In fact, in some instances, $T_{\text{max}}$ values are very tight, which makes finding good feasible solutions very difficult, specially when the number of required arcs that the solution has to traverse obligatorily is large (in instances with $p = 0.5$, for example). Looking at the results for class D (see Table III), we observe that this class contains more difficult instances. The branch-and-cut algorithm is more sensitive to the number of vehicles. In fact, the number of instances solved to optimality decreases when $K$ increases. MAT is instead more sensitive to parameter $p$. Note also that the average gap calculated with respect to the optimal solution is in some cases much lower than the one calculated with respect to the upper bound. This indicates that the upper bound may be quite far from the optimal solution. The same observations apply to class G (see IV).

Finally, in Table V we summarize the results by presenting the aggregated average gaps classified by class of instances, number of vehicles, parameter $p$ and finally the total average gap on all instances. We notice that parameter $p$ has a higher influence than the number of vehicles, where instances with $p = 0.5$ are the most difficult ones. The explanation is the same we mentioned above for the instance of class R. For these instances the solution space is much narrower than for the cases with $p = 0$ or $p = 0.25$. Thus, when the algorithm is not able to find the optimal solution, it tends to give large errors.

| Set   | # inst | $|V|$ | $|A|$ | $|A_R|$ | $|A_P|$ | $|A_R|$ | $|A_P|$ | $|A_R|$ | $|A_P|$ |
|-------|--------|------|------|--------|--------|--------|--------|--------|--------|
| R30   | 5      | 11-18| 42-134| 0      | 7-11   | 1-3    | 4-9    | 3-6    | 6-2    |
| R40   | 5      | 13-25| 68-266| 0      | 8-18   | 3-5    | 3-15   | 4-8    | 1-11   |
| R50   | 5      | 19-27| 166-296| 0      | 13-20  | 0-7    | 11-17  | 4-11   | 8-10   |
| D36   | 9      | 17-36| 96-270| 0      | 10-38  | 2-10   | 6-30   | 6-20   | 4-23   |
| D64   | 9      | 37-62| 264-482| 0      | 27-75  | 4-21   | 22-54  | 11-38  | 15-37  |
| D100  | 9      | 68-100| 544-846| 0      | 50-121 | 9-28   | 37-95  | 26-64  | 20-70  |
| G36   | 9      | 18-35| 54-120| 0      | 11-35  | 1-11   | 7-28   | 6-18   | 3-19   |
| G64   | 9      | 34-62| 128-228| 0      | 24-68  | 3-22   | 20-50  | 10-15  | 12-38  |
| G100  | 9      | 60-100| 246-394| 0      | 41-113 | 8-25   | 33-91  | 19-57  | 20-64  |

Table I: Characteristics of the TOARP instances
### Table II: Computational results with the sets of instances R30, R40, and R50

<table>
<thead>
<tr>
<th></th>
<th>R30</th>
<th>R40</th>
<th>R50</th>
<th>R30</th>
<th>R40</th>
<th>R50</th>
<th>R30</th>
<th>R40</th>
<th>R50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = 0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td># opt</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.79</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.79</td>
</tr>
<tr>
<td>$K=2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p = 0.25$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td># opt</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$p = 0.5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td># opt</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Table III: Computational results with the sets of instances D36, D64, and D100

<table>
<thead>
<tr>
<th></th>
<th>D36</th>
<th>D64</th>
<th>D100</th>
<th>D36</th>
<th>D64</th>
<th>D100</th>
<th>D36</th>
<th>D64</th>
<th>D100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = 0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td># opt</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.48</td>
<td>2.58</td>
<td>0.00</td>
<td>0.26</td>
<td>2.92</td>
<td>0.00</td>
<td>1.10</td>
<td>4.71</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.48</td>
<td>2.58</td>
<td>0.00</td>
<td>0.26</td>
<td>2.92</td>
<td>0.00</td>
<td>1.10</td>
<td>4.71</td>
</tr>
<tr>
<td>$K=2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p = 0.25$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td># opt</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.10</td>
<td>3.31</td>
<td>0.08</td>
<td>0.42</td>
<td>4.14</td>
<td>0.00</td>
<td>2.05</td>
<td>20.45</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.10</td>
<td>0.19</td>
<td>0.08</td>
<td>0.42</td>
<td>1.33</td>
<td>0.00</td>
<td>2.05</td>
<td>8.91</td>
</tr>
<tr>
<td>$p = 0.5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># solved</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td># opt</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>1.42</td>
<td>4.58</td>
<td>0.00</td>
<td>0.68</td>
<td>3.02</td>
<td>0.00</td>
<td>1.21</td>
<td>7.53</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
<td>0.15</td>
<td>0.05</td>
<td>0.00</td>
<td>0.30</td>
<td>2.53</td>
</tr>
<tr>
<td>( p = 0 )</td>
<td>( p = 0.25 )</td>
<td>( p = 0.5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( K=2 )</td>
<td>( K=3 )</td>
<td>( K=4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
</tr>
</thead>
<tbody>
<tr>
<td># solved</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td># opt</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.63</td>
<td>1.56</td>
<td>0.00</td>
<td>1.58</td>
<td>1.53</td>
<td>0.00</td>
<td>0.00</td>
<td>2.62</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.63</td>
<td>1.35</td>
<td>0.00</td>
<td>1.58</td>
<td>1.21</td>
<td>0.00</td>
<td>0.00</td>
<td>2.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
</tr>
</thead>
<tbody>
<tr>
<td># solved</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td># opt</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.00</td>
<td>0.25</td>
<td>3.27</td>
<td>0.00</td>
<td>0.00</td>
<td>3.33</td>
<td>0.00</td>
<td>1.18</td>
<td>5.04</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.25</td>
<td>1.62</td>
<td>0.00</td>
<td>0.00</td>
<td>1.44</td>
<td>0.00</td>
<td>1.18</td>
<td>4.44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
<th>G36</th>
<th>G64</th>
<th>G100</th>
</tr>
</thead>
<tbody>
<tr>
<td># solved</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td># opt</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Av. Gap</td>
<td>0.38</td>
<td>4.72</td>
<td>8.53</td>
<td>0.53</td>
<td>0.00</td>
<td>6.38</td>
<td>1.96</td>
<td>0.00</td>
<td>10.87</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.06</td>
<td>0.00</td>
<td>0.00</td>
<td>1.52</td>
</tr>
</tbody>
</table>

**Table IV: Computational results with the sets of instances G36, G64, and G100**

<table>
<thead>
<tr>
<th></th>
<th>Class R</th>
<th>Class D</th>
<th>Class G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Av. Gap</td>
<td>0.16</td>
<td>2.26</td>
<td>2.01</td>
</tr>
<tr>
<td>Av. Gap*</td>
<td>0.16</td>
<td>1.05</td>
<td>0.66</td>
</tr>
</tbody>
</table>

**Table V: Average results**
Conclusions

The matheuristic proposed for the solution of the Team Orienteering Arc Routing Problem is shown to be efficient as it has produced an average error of 0.64% on a large number of benchmark instances for which the optimal solution is known. It finds the optimal solutions on all, except one, instances of one of the three classes of tested instances (with up to 27 vertices and 296 arcs). The effectiveness of the algorithm is the result of the use of ILP models within a tabu search scheme that allow us to explore some parts of the solution space in depth, taking advantage of the power of commercial software. Moreover, the solution value given by the matheuristic has been used as initial upper bound in the branch-and-cut algorithm proposed in [5] and proved to be effective in improving the efficiency of the solution procedure.

Recent advances in the design of exact methods and heuristics make it possible to achieve good results on arc routing problems, a class of problems that is still largely unexplored if compared to node routing problems.

Acknowledgements: Ángel Corberán, Isaac Plana and José M. Sanchis wish to thank the Ministerio de Educación, Cultura y Deporte of Spain (projects MTM2009-14039-C06-02 and MTM2012-36163-C06-02) for its support.

References


