A Matheuristic for the Team 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, Università di Brescia, Italy*

[(2)] *Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain*

[(3)] *Departamento de Matemáticas para la Economía y la Empresa, Universidad de Valencia, Spain*

[(4)] *Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Spain*

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

October 1, 2014

Abstract

In the Team Orienteering Arc Routing Problem (TOARP) the potential customers are located on the arcs of a directed 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 is known. The matheuristic finds the optimal solutions on all, except one, instances of one of the four 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 0.67%.

Keywords

Team Orienteering Problem, Arc Routing Problem, Routing Problems with Profits, Matheuristic.

*corresponding author

1 Introduction

In arc routing problems customers are located on arcs. The basic problems of this class are the Chinese Postman Problem (CPP), where all edges or arcs have to be traversed, and the Rural Postman Problem (RPP), where only some edges or arcs are required to be traversed, and routes of minimum cost have to be identified. For a formal definition of the CPP and RPP the reader is referred to the book edited by Dror [10]. There are several applications of arc routing problems where the edges or arcs to be traversed 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 traverse 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 traverse 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 [12], [20] and [7]). The arc routing counterpart of this class is called *arc routing problems with profits*. A recent survey on arc routing problems with profits can be found in [5]. We refer the reader to this survey for the description of the problems that have been studied till now in the literature. It is important to mention that, as pointed out in [5], the number of papers dealing with arc routing problems with profits is still very limited, especially if compared with the node routing counterpart. More specifically, the contributions related to the study of problems dealing with more than one vehicle are very few. In fact, the only problems that have been studied (apart from the TOARP that is studied in this paper) are the Profitable Arc Tour Problem (PATP) and the Undirected Capacitated Arc Routing Problem with Profits (UCARPP). The PATP was introduced in [13] and is the problem of finding a set of routes maximizing the difference between the total collected profit and the travel cost such that the travel time of each route does not exceed a given threshold. No limit on the number of routes is given. In [13] an exact algorithm is proposed based on the branch-and-price scheme while in [11] different heuristic algorithms are developed. The UCARPP was introduced in [3] and is the problem of finding the set of routes that maximizes the total collected profit in such a way that each route satisfies capacity and maximum

duration constraints. In [3] an exact algorithm and different heuristics are proposed. More recently, the same problem was addressed in [21] where a new and effective heuristic algorithm is presented.

A well studied problem in the class of routing problems with profits 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 [1]. 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 is 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 customers place orders consisting of requests of transportation services from an origin to a destination. Each transportation service requires a full truck going from the corresponding origin location to the corresponding destination. These services can be represented as arcs of a graph that have to be traversed in order to satisfy the corresponding customer requests. 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 [1] an extended polyhedral study for the TOARP is presented. The proposed branch-and-cut algorithm solves instances with up to 100 vertices, 800 arcs, and 4 vehicles to optimality and makes use of the solutions provided by the heuristic described in this paper.

We address the heuristic solution of the TOARP by means of an algorithm that combines a tabu search, to escape from local optima, a diversification phase and the optimal solution of integer linear programming (ILP) models to intensify the search in some areas of the solution space. The combination of a heuristic or metaheuristic scheme with mixed integer linear programming (MILP) models has been recently explored by several authors. A survey is presented by Ball [8] for combinatorial optimization problems in general while, more recently, in [6] a survey focused on routing problems can be found. These heuristics are named in different ways. In the survey [8] they are simply called *heuristics based on mathematical programming*, in other cases (see [19]) they go under the generic name of *hybrid heuristics*. The name *matheuristic* was created ad hoc for this class of heuristics (see [17]) and this is the name we will use in this

paper. The main contribution of this paper is the design of a matheuristic for the TOARP where we exploit the benefit of using ILP models in the searching phase. We developed a solution algorithm which combines heuristic operators with the exact solution of different ILP models. These models can be easily implemented and are powerful tools to intensify the search around a promising solution, while typically standard heuristic operators are not as effective or have to be adapted to the problem structure, thus leading to very specialized solution methods. We performed tests on a large set of benchmark and randomly generated instances. The matheuristic finds the optimal solution on 78% of the instances for which the optimal solution is known and provides an average error with respect to the optimal solution of 0.67%.

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 cost or travel time is associated with each arc. Only some of the arcs represent customers. Some customers have to be 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 in such a way 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 travel time c_a is associated with each arc $a \in A$. Some arcs represent customers. The set $A_R \subseteq A$ represents customers that have to be served, whereas $A_P \subseteq A$ represents the set of profitable customers. A nonnegative profit s_a is associated with each arc $a \in A_P$.

A fleet of K vehicles is available. The route of each vehicle cannot exceed a maximum time duration T_{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. A mathematical programming formulation for the TOARP can be found in [1].

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, we redefine A_P as $A_P \cup A_R$. Moreover, the set of arcs A is completed by inserting all arcs between every pair of profitable arcs, plus the depot. Thus, if there is not an arc which link the head of a profitable arc (or the depot) with the tail of another profitable arc (or the depot), then we insert it in A. The cost of the inserted arcs is equal to that of the shortest path.

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

3.1 Notation 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 by 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 defined as a set of routes such that each profitable arc is served by exactly one route. A solution s is said to be *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 organized in routes. The reason for keeping 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 the K most profitable routes only, it turned out to be beneficial to the efficiency of the heuristic. This solution structure was used effectively in the solution of other routing problems with profits like UCARPP (see [3]), the TOP (see [4]), the Capacitated Team Orienteering Problem and the Capacitated Profitable Tour Problem (see [2]).

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 $r_a(s)$ and a profitable arc a served by $r_a(s)$, we denote $r_a(s) - a$ the route obtained from $r_a(s)$ by removing a. The insertion of an arc a into a route r is performed by means of the ADD operator introduced by Hertz et al. [14] for the Undirected Capacitated Arc Routing Problem (UCARP), adapted to the case of a directed graph. The removal of arc a from route $r_a(s)$ is performed by means of the DROP operator by Hertz et al. [14] for the UCARP, adapted to the case of a directed graph.

3.2 MAT scheme

MAT is a matheuristic which combines a tabu search and a diversification phase with the exact solution of ILP models. A tabu search phase is carried out to guide the search in the neighborhood of the current solution. The ILP models are used to intensify the search with the aim to improve the solution found during the tabu search phase. At the end of the tabu search phase, a diversification step, called *jump*, is performed to move the search to a completely different part of the solution space. A general scheme of MAT is given in Figure 2.

INITIAL SOLUTION. Let s be the generated solution.

$k := k_{max}$.

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

JUMP (k, s): Generate a new solution s' by performing k moves on solution s.

$s \leftarrow s'$.

INTERNAL TABU SEARCH (s): Apply a tabu search to improve upon solution s. Let s' be the generated solution.

$s \leftarrow s'$.

If s is better than s_{best} **then**

ROUTE IMPROVEMENT (s). Let s_{best} be the generated solution.

$s \leftarrow s_{best}$.

$k := k_{max}$.

Else

$k := k - 1$.

If $k = 0$ **then**

$k := k_{max}$.

End If

End If

INTENSIFICATION PHASE (s). Intensify the search around solution s. Let s' be the generated solution.

$s \leftarrow s'$.

If s is better than s_{best} **then**

$s_{best} \leftarrow s$.

End If

End While

Figure 2: MAT: A matheuristic for the TOARP

We now describe in details the procedures that compose MAT.

3.3 Initial solution

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

3.4 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 in 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 $r_a(s)$ and insert it in r'. If $T(r') > T_{max}$, 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 $r'_a(s)$ and insert it in a route $r'' \in R_N(s)$.

The *sequence move* may be seen as a combination of 1-moves or swap-moves. The sequence move performs a more intensive search around the current solution than the search performed through the application of the *1-move* and the *swap-move* only. However, applying the sequence move at every iteration may be too cumbersome. Thus, it is performed only every 5 iterations.

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

$$l = \sqrt{\lambda\sigma/4} + |A_P| * \sqrt{K}/8, \tag{1}$$

where $\lambda = |V| * |R|$, with R the set of routes in the initial solution and σ a random number in $[0, 1]$. This formula links the number of tabu iterations to the

number of vertices and profitable arcs, and to the number of routes. Parameter σ is introduced in order to have a certain variability in the number of tabu iterations.

A hierarchical function is used to evaluate the solution quality which considers the following terms, in the order in which we describe them: the total profit $S(R_P(s))$ of the routes in $R_P(s)$, the total duration $\sum_{r \in R_P(s)} T(r)$ of the routes in $R_P(s)$, the number of non empty routes in s and, finally, the total duration $\sum_{r \in R_N(s)} T(r)$ of the routes in $R_N(s)$.

The tabu search phase terminates after 400 iterations without improvement. This value allows us to keep this phase quite short and to perform a large number of jumps to diversify the search.

3.5 Route improvement (s)

The ROUTE IMPROVEMENT (s) phase is aimed at optimizing the order of traversal of the arcs in the routes in s. For each route of the current best solution s, this procedure is aimed at reducing the corresponding duration and, at the same time, serving all profitable arcs currently served by the route. This phase is particularly important when the routes are long, i.e., when they traverse a large number of arcs, which is the case in most of the tested instances.

The procedure works as follows. Given a new best solution s, we first reduce the time duration of the routes in $R_P(s)$ to increase the probability of inserting additional profitable arcs and then apply the *sequence move*. To reduce the time duration of the routes in s, for each route r in $R_P(s)$ we solve an Asymmetric Traveling Salesman Problem (ATSP) where each vertex of the graph corresponds to a sequence of profitable arcs which are served consecutively 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 last profitable arc of the sequence represented by vertex i and the tail of the first profitable arc of the sequence represented by vertex j. An example of an ATSP instance generated by a TOARP route is shown in Figure 1 where the square vertex is the depot while the arcs in bold are the profitable arcs served by the route. Numbers close to each arc represent the corresponding cost.

The ATSP is solved using the Miller-Tucker-Zemlin formulation [18], which has a polynomial number of constraints. After that, the *sequence move* is applied iteratively as long as it improves the current solution.

A maximum time of 30 seconds is assigned to the solution of the ATSP.

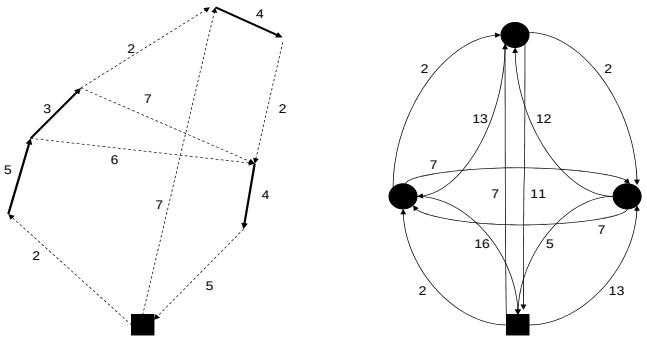

Figure 1: Transforming a TOARP route into an ATSP instance

We decided to give such a short computing time as the procedure is applied very often. As the routes are often formed by a large number of profitable arcs, it may happen that the optimal solution of the ATSP is not found within 30 seconds. In this case, the best feasible solution found is accepted. We tested also the use of the heuristic algorithm of Lin and Kernighan [16] to solve the ATSP. However, this leads to worse results.

3.6 Intensification phase (s)

The Intensification phase (s) is applied once the Internal tabu search (s) phase is terminated and before performing a jump. The aim of the jump is to destroy the current solution and to 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 provided in Section 3.7. The aim of the Intensification phase (s) is to intensively exploit the possibility of improving the current best solution.

The Intensification phase (s) consists in the solution of two ILP models which differ in terms of aim and structure. The first ILP model, which we call Route selection model, aims at selecting the K most profitable routes among the set $\mathcal{R}$ of all the routes generated so far. Thus, the solution space is defined by the entire set $\mathcal{R}$ of routes generated by the algorithm (which may be quite large). The second model, which we call Profit improvement model,

focuses on the best solution s found so far and aims at increasing the profit collected by the routes in $R_P(s)$ by inserting profitable arcs or sequences of profitable arcs and removing less profitable arcs.

Let us now describe the two models in more detail. Let parameter b_{ar} be 1 if the profitable arc a is traversed by route $r \in \mathcal{R}$, and 0 otherwise. ROUTE SELECTION makes use of binary variables z_a, where $z_a = 1$ if profitable arc a is served, and 0 otherwise, and variables x_r, where $x_r = 1$ if route $r \in \mathcal{R}$ is used, and 0 otherwise. The ROUTE SELECTION model is as follows:

$$\text{Maximize} \qquad \sum_{a \in A_P} s_a z_a$$

$$\sum_{r \in \mathcal{R}} b_{ar} x_r \geq z_a \qquad a \in A_P \tag{2}$$

$$\sum_{r \in \mathcal{R}} x_r \leq K \tag{3}$$

$$x_r \in \{0,1\} \qquad r \in \mathcal{R} \tag{4}$$

$$z_a \in \{0,1\} \qquad a \in A_P. \tag{5}$$

The objective function aims at maximizing the total collected profit. Constraints (2) establish that at least a route must be chosen that traverses a served arc while (3) limits the number of routes that can be selected.

The PROFIT IMPROVEMENT model tries to insert in $R_P(s)$ single profitable arcs or pairs of profitable arcs. For each profitable arc a, the insertion cost of a in r is calculated as the detour cost for adding a to r. The cost of inserting two profitable arcs a and a' in route r is calculated as the minimum between the cost of going from r to a, from a to a' and from a' to r and the cost of going from r to a', from a' to a and from a to r. The insertion cost of the pair of arcs a and a' may be in general lower than the sum of the insertion costs of the single arcs a and a', and this happens especially when a and a' are close to each other. This is the reason why we consider explicitly the insertion of pairs of 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

$$T = \frac{T_{max} K}{|L(R_P(s))|},$$

which gives the average time available to serve the profitable arcs in $R_P(s)$. We did not consider longer sequences to avoid an excessive increase of the computing time.

Let $\mathcal{S}$ be the set of all sets composed by single profitable arcs and pairs of profitable arcs with a distance lower than T. We denote by $S(i)$ the profit of set $i \in \mathcal{S}$. Let Γ_{ir} be the cost of inserting set $i \in \mathcal{S}$ in route $r \in R_P(s)$ and Δ_{ar} be the 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). The PROFIT IMPROVEMENT model 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. The formulation is the following:

$$\text{Maximize} \quad \sum_{i \in \mathcal{S}} \sum_{r \in R_P(s)} \Big(S(i) v_{ir} + \sum_{a \in A_P} s_a (f_{air} z_{air}) \Big) - \sum_{r \in R_P(s)} \sum_{a \in A_P} s_a w_{ar}$$

$$T(r) + \sum_{i \in \mathcal{S}} \Gamma_{ir} v_{ir} - \sum_{a \in A_P} \Delta_{ar} w_{ar} \leq T_{max} \qquad r \in R_P(s) \tag{6}$$

$$v_{ir} \leq 1 - g_{ai} b_{ar} \qquad a \in A_P, i \in \mathcal{S}, r \in R_P(s) \tag{7}$$

$$w_{ar} \leq b_{ar} \qquad a \in A_P, r \in R_P(s) \tag{8}$$

$$z_{air} \leq f_{air} v_{ir} \qquad a \in A_P, i \in \mathcal{S}, r \in R_P(s) \tag{9}$$

$$\sum_{r \in R_P(s)} (b_{ar} - w_{ar} + \sum_{i \in \mathcal{S}} (g_{ai} v_{ir} + f_{air} z_{air})) \leq 1 \qquad a \in A_P \tag{10}$$

$$\sum_{a \in A_P} w_{ar} + \sum_{i \in \mathcal{S}} v_{ir} \leq \Theta \qquad r \in R_P(s) \tag{11}$$

$$v_{ir} \in \{0,1\} \qquad i \in \mathcal{S}, r \in R_P(s) \tag{12}$$

$$w_{ar} \in \{0,1\} \qquad a \in A_P, r \in R_P(s) \tag{13}$$

$$z_{air} \in \{0,1\} \qquad a \in A_P, i \in \mathcal{S}, r \in R_P(s). \tag{14}$$

The objective function aims at maximizing the profit of the routes in $R_P(s)$. The left-hand side of constraints (6) gives an estimation of the cost of each new route r which must be lower than T_{max}. Constraints (7) 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 (8) an arc a can be removed from route r only if r serves a. Inequalities (9) impose that variable z_{air} is set to 1 only if sequence i is inserted in route r and arc a is on the shortest path that links i with r. (10) guarantee that each profitable arc is served at most once. Finally, with (11) we impose that a limited number Θ of arcs or sets of arcs can be removed or inserted from a route.

The limitation set in constraints (11) is applied to avoid a poor estimation of the cost of the new route in constraint (6). In fact, the estimation of the cost of the route made in constraint (6), i.e., $T(r) + \sum_{i \in \mathcal{S}} \Gamma_{ir} v_{ir} - \sum_{a \in A_P} \Delta_{ar} w_{ar}$, is an approximation of the cost of the new route. Indeed, if we insert and/or remove more than one arc of the same route, then the cost of the new 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 traversed consecutively in route r (meaning that no other profitable arc is served between the two), with a traversed before a'. Then the cost of the arcs connecting the head of a with the tail of a' is accounted for both in Δ_{ar} and in $\Delta_{a'r}$, and thus twice in the left-hand side of (6). As an example, consider the route depicted in Figure 2. The savings obtained by removing a and a' are both equal to 5, thus summing them up we obtain a total saving of 10. However, when removing both of them simultaneously, we obtain a saving of 5.

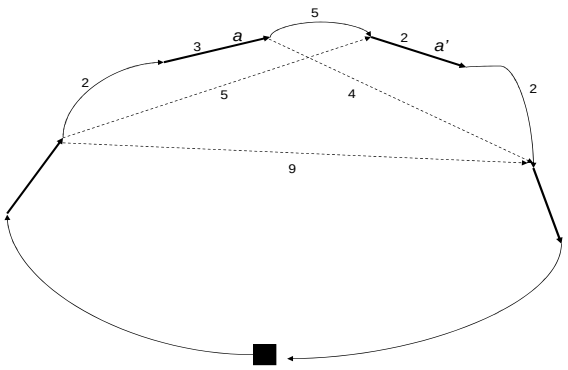

Figure 2: A poor approximation of the cost of the new route

W set the value of Θ to 5. This value is sufficient to allow a deep search of the solution space around solution s and, on the other hand, to prevent from a poor approximation of the cost of each route.

Note that, because of constraints (6), the solution obtained by the PROFIT IMPROVEMENT model may be infeasible. In this case, we randomly remove arcs from infeasible routes until each route becomes feasible. The removed arcs are inserted in existing routes in $R_N(s)$, if this is feasible, or in new routes.

The PROFIT IMPROVEMENT model is solved only if the ROUTE SELECTION model is able to improve the current best solution. If this is not the case, the current best solution has already been processed by the ROUTE IMPROVEMENT (s) procedure and thus the probability of improving this solution further through PROFIT IMPROVEMENT is low. If instead a new best solution is found by ROUTE SELECTION, then PROFIT IMPROVEMENT makes a deep search around this solution to try to increase the corresponding profit.

3.7 Jump (k, s)

The JUMP (k, s) procedure is performed after the intensification phase. 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 $S(U) \leq S(W)$. The larger the value of k, the more different the new solution is with respect to the previous one.

When we move the arcs from $R_N(s)$ to $R_P(s)$, we disregard the T_{max} duration constraint and we insert each arc in the route that leads to the cheapest insertion cost. Consequently, the solution obtained after a jump may be infeasible. In order to recover feasibility, we first optimize the length of each route by solving an ATSP as described in Section 3.5 and then apply 1-moves and swap-moves on 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 randomly remove arcs from infeasible routes until each route becomes feasible. The removed arcs are inserted in existing routes in $R_N(s)$, if this is feasible, or in new routes.

At the end of the procedure that recovers the feasibility of the routes in $R_P(s)$ we may obtain a solution of very poor quality which may constitute a bad starting point for the following internal tabu search phase. This has a very bad impact on the quality of the solutions found by the internal tabu search as a short computing time is allowed to it. Thus, in order to improve the starting solution, a further ILP model, which we call FAST PROFIT IMPROVEMENT, is solved on the solution generated after the jump. FAST PROFIT IMPROVEMENT is a simplification of PROFIT IMPROVEMENT, as applying PROFIT IMPROVEMENT at each jump proved to be too cumbersome. As done by PROFIT IMPROVEMENT, FAST PROFIT IMPROVEMENT 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 PROFIT IMPROVEMENT, in FAST PROFIT IMPROVEMENT it is possible to only insert or remove single profitable arcs from the routes in $R_P(s)$.

For the sake of completeness and clarity, we report the formulation of FAST PROFIT IMPROVEMENT even if it is very similar to the one of PROFIT IMPROVEMENT. With a similar notation as in PROFIT IMPROVEMENT, let b_{ar} be equal to 1 if profitable arc a is currently served by route $r \in R_P(s)$, Γ_{ar} be the cost of inserting arc $a \in A_P$ in route $r \in R_P(s)$ and Δ_{ar} be the saving gained if arc a is removed from route $r \in R_P(s)$. FAST PROFIT IMPROVEMENT 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. FAST PROFIT IMPROVEMENT is formulated as follows:

$$\text{Maximize} \qquad \sum_{a \in A_P} \sum_{r \in R_P(s)} (s_a v_{ar} - s_a w_{ar})$$

s.t.:

$$T(r) + \sum_{a \in A_P} (\Gamma_{ar} v_{ar} - \Delta_{ar} w_{ar}) \leq T_{max} \qquad r \in R_P(s) \tag{15}$$

$$v_{ar} \leq 1 - b_{ar} \qquad \forall a \in A_P, \quad r \in R_P(s) \tag{16}$$

$$w_{ar} \leq b_{ar} \qquad \forall a \in A_P, \quad r \in R_P(s) \tag{17}$$

$$\sum_{r \in R_P(s)} (b_{ar} - w_{ar} + v_{ar}) \leq 1 \qquad a \in A_P \tag{18}$$

$$\sum_{a \in A_P} (w_{ar} + v_{ar}) \leq \Theta \qquad r \in R_P(s) \tag{19}$$

$$v_{ar} \in \{0, 1\} \qquad a \in A_P, r \in R_P(s) \tag{20}$$

$$w_{ar} \in \{0, 1\} \qquad a \in A_P, r \in R_P(s). \tag{21}$$

The objective function and the constraints have a similar meaning as in PROFIT IMPROVEMENT. The differences are related to the fact that we consider single arcs to be inserted in the routes and not pair of arcs. A further difference is the absence of variables z_{air} which leads to a different formulation of constraints (6) and (10) and to the absence of constraints (14).

In FAST PROFIT IMPROVEMENT we fix the value of Θ to 3 in order to reduce the computing time. In fact, a larger value of Θ leads to a wider solution space and thus the problem becomes harder to solve. Moreover, when Θ increases, the risk of a poor approximation of the cost of the routes increases and, at the same time, the risk to obtain infeasible solutions increases. Given that FAST PROFIT

IMPROVEMENT is solved frequently, we prefer to obtain a feasible solution in a fast way.

Infeasibility is eventually recovered by randomly removing arcs from infeasible routes until each route becomes feasible. The removed arcs are inserted in existing routes in $R_N(s)$, if this is feasible, or in new routes.

Finally, note that, even if FAST PROFIT IMPROVEMENT is applied, the solution obtained after a jump may be of poor quality and starting a tabu search phase from a 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 $z_{best}\beta$, where z_{best} is the value of the best solution found so far. β is initialized to 0.9. If this is not the case, a new jump is performed. This procedure is aimed at guaranteeing a good quality of the starting solution of the internal tabu search phase. After 5 consecutive trials for which the value of the solution is lower than $z_{best}\beta$, the value of β is decreased to 0.9β. In fact, it may happen that the threshold of $z_{best}\beta$ is too difficult to reach. This is why we decrease the value of β after 5 unsuccessful iterations.

4 Computational results

MAT was tested on the set of instances proposed in [1]. These instances are generated from three classes of RPP benchmark instances proposed by Hertz et al. [15]. In the first class, the R class, the edges are generated randomly in the plane. In the second class, the D class, the edges define a graph where all the vertices have degree 4. In the third class, the G class, the edges have all cost 1 and define a uniform grid. 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 neither required nor profitable. The profit associated with the traversal of a profitable arc is defined as its cost, which corresponds to assuming that the profit gained by serving an arc is proportional to the traversal cost. Each of these classes of instances is divided in three sets whose characteristics are reported in Table I.

The instances of classes R, D and G are based on undirected graphs and, thus, are such that if there is an arc from i to j, then there is also an arc from j

to i with the same cost. Given that the TOARP is defined on a directed graph, we generated a new class of instances where some arcs exist in one direction but not in the other one. Moreover, in this new class of instances, profits are not related to the arc costs but are generated randomly. The number of vertices was set equal to 50 and the vertices were randomly generated in the $1,000 \times 1,000$ square. Then, for each vertex i, arcs to the three vertices closest to i are added to the arc set A. Moreover, to guarantee the strong connectivity of the resulting graph, the arcs in three different Hamiltonian cycles are also added to A. Arc costs are defined as the Euclidean distances. Each arc $a \in A$ is randomly included in $A_R \cup A_P$ with probability 0.2. We generated 10 such instances with $p = 0$ and 10 instances with $p = 0.5$. Profits of the arcs in A_P were randomly generated in the interval $[100; 500]$. We call this class of instances the T50 instances.

For each one of the previous instances, we generated three instances with $K = 2$, 3 and 4. The value of T_{max} was set in such a way that a feasible solution exists while there is no solution which traverses all profitable and required arcs. This value was found by solving the minmax K-vehicles Directed Rural Postman Problem with the algorithm proposed in [9], as explained in [1].

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. To calculate the optimal solution of the ATSP, Route selection, Profit improvement and fast profit improvement we used CPLEX 12.2.0. The maximum time allowed for the solution of each ATSP was set to 30 seconds while 30 minutes were allowed for the solution of Route selection, Profit improvement and Fast profit improvement. This choice is due to the fact that the ATSP is solved very often and thus we decided to allow a short computing time. Route selection, Profit improvement and Fast profit improvement are instead called less frequently and, in addition, they are typically solved much faster than the ATSP. Thus, we decided not to insert a maximum computing time for them and to fix it equal to the maximum time allowed for MAT.

Classes R, D and G are composed by 207 instances and the optimal solution is known for 204 instances with 2 vehicles and for 188 and 157 instances with 3 and 4 vehicles, respectively. For the T50 instances, the optimal solution is known for 19 instances out of 20 with 2 vehicles, for 13 out of 20 with 3 vehicles and for 4 out of 20 with 4 vehicles. The optimal solution was obtained through

the branch-and-cut algorithm proposed in [1].

Tables II-V summarize the results. Table II refers to the instances of class R, Table III to class D, Table IV to class G, and Table V to class T50. 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 [1] (# solved), the number of optimal solutions found by MAT (# opt), the average and maximum percentage gap of the solution found by MAT with respect to the upper bound found by the branch-and-cut algorithm (Av. and Max Gap) and the average and maximum percentage gap calculated only on the instances where the optimal solution is available (Av. and Max Gap*).

From Tables II - V one can note that the simplest class of instances, both for the exact algorithm and the heuristic, 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 determines the average error of 5.79% over five instances 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, the T_{max} value is very tight, which makes finding good feasible solutions very difficult. 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 sensitive to the number of vehicles as the number of instances solved to optimality decreases when K increases. MAT is instead more sensitive to parameter p. 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 Table IV). The instances of class T50 (Table V) are difficult to solve to optimality especially for $K = 4$ but MAT gives on average good quality solutions.

The values of the maximum gaps are quite high in some cases. We now focus on the instances for which MAT generated large errors and analyze the reasons for such a behavior. We analyze the instances solved to optimality as in the other cases the quality of the upper bound may be very poor. We just mention that, for the instance of Table IV which presents a value of Max Gap of 44.61%, the value of the solution of MAT corresponds to the best feasible

solution found by the branch-and-cut algorithm.

Table VI presents an analysis of the structure of the optimal solution of the instances for which the value of the solution obtained through MAT is greater by more than 10% with respect to the value of the optimal solution. The first four columns report data on the instance: name, set, number of vehicles and parameter p. The fifth column reports the percentage gap between the MAT solution and the value of the optimal solution. The last three columns present the average number of arcs traversed by a route in the optimal solution (# traversed arcs per route), the average number of profitable and required arcs served in a route (# required and profitable arcs per route), and the ratio between the total duration of the routes with respect to the maximum allowed duration ($\frac{\sum_{r \in R_p(s)} T(r)}{kT_{max}}$).

The larger number of instances for which MAT generated an error greater than 10% comes from set D100 with $p = 0.5$. The last column of the table shows that the duration constraint is extremely binding for all these instances. Moreover, with the only exception of instance 'hertzr16', all remaining instances have long routes serving a large number of profitable and required arcs. This is a feature that, combined with a binding duration constraint, makes these instances very difficult to solve heuristically. Concerning instance 'hertzr16', we have performed different tests by changing the parameters used in MAT and we have always obtained two solution values: either the optimal one or the one which is 29% worse. Thus, we believe that the solution we report is the second best.

In Table VII we summarize the results related to the average gaps and number of optimal solutions found aggregated by class of instances, number of vehicles, and parameter p. The last row of the table refers to the total average gap and total number of optimal solutions found on all instances. We notice that when the value of p increases, the instances seem to be more difficult to solve heuristically.

Finally, in Table VIII we report the results of a computational study we made on the set of instances considered in Table VI. The aim of this study is to analyze the effectiveness of the different components of MAT and the tuning of the parameters. We perform the analysis on the following procedures/parameters used in MAT: number of tabu iterations, intensification phase, route improvement phase, sequence move and value of parameter Θ in constraint (11). The reason of our choice is that we believe that these are the

procedures/parameters that influence the most the performance of MAT. We report the percentage gap of the solutions produced by MAT with respect to the optimal solution for the different settings of MAT. In particular, in the second column we report the gaps obtained with the setting described in the paper. The third column refers to the case where the number of tabu iterations, as calculated in (1), is halved. The fourth and the fifth columns report the results for the case where the intensification phase described in Section 3.6 and the route improvement described in Section 3.5 are not performed, respectively. The last two columns report the results obtained when no sequence move is performed and when the value of the parameter Θ in constraint (11) is set to 5, respectively. The results show that there is no setting that is dominant with respect to the others. During the preliminary tests, we also observed that, with a different setting, the set of instances that show a large gap changes. The setting used in our tests is the one that guarantees to always find a feasible solution and to avoid extremely bad solutions.

Conclusions

The matheuristic proposed for the solution of the Team Orienteering Arc Routing Problem generates an average error of 0.67% and finds the optimal solution on 78% of the instances for which the optimal solution is known. The effectiveness of the algorithm is the result of the use of ILP models combined with a tabu search and a diversification phase that allows us to explore different parts of the solution space in depth, taking advantage of the power of commercial MILP solvers.

The solution value generated by the matheuristic was used as initial upper bound in the branch-and-cut algorithm proposed in [1] and proved to be effective in improving its efficiency.

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: The authors wish to thank two anonymous referees for their useful comments which helped them improve a previous version of the paper.

Ángel Corberán, Isaac Plana and José M. Sanchis wish to thank the Ministerio de Economía y Competitividad (project MTM2012-36163-C06-02) of Spain

and the Generalitat Valenciana (project GVPROMETEO2013-049) for their support.

References

[1] C. Archetti, Á. Corberán, I. Plana, J.M. Sanchis and M.G. Speranza (2012), The team orienteering arc routing problem. *Transportation Science*, to appear, doi:10.1287/trsc.2013.0484.

[2] C. Archetti, D. Feillet, A. Hertz and M.G. Speranza (2009), The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society* 60, 831-842.

[3] C. Archetti, D. Feillet, A. Hertz and M.G. Speranza (2010), The undirected capacitated arc routing problem with profits. *Computers & Operations Research* 37, 1860-1869.

[4] C. Archetti, A. Hertz and M.G. Speranza (2007), Metaheuristics for the team orienteering problem. *Journal of Heuristics* 13, 49-76.

[5] C. Archetti and M.G. Speranza (2013), Arc routing problems with profits. Working paper WPDEM 2013/2, Department of Economics and Management, University of Brescia.

[6] C. Archetti and M.G. Speranza (2013), A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, doi:10.1007/s13675-014-0030-7.

[7] C. Archetti, M.G. Speranza and D. Vigo (2013), Vehicle routing problems with profits. Working paper WPDEM 2013/3, Department of Economics and Management, University of Brescia.

[8] M.O. Ball (2011), Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science* 16, 21-38.

[9] E. Benavent, Á. Corberán, I. Plana and J.M. Sanchis (2011), New facets and an enhanced branch-and-cut for the min-max k-vehicles windy rural postman problem. *Networks* 58, 255-272.

[10] M. Dror (2000), *Arc Routing. Theory, Solutions and Applications.* Kluwer Academic Publishers, Boston.

[11] J. Euchi and H. Chabchoub (2011), Hybrid metaheuristics for the profitable arc tour problem. *Journal of the Operational Research Society* 62, 2013-2022.

[12] D. Feillet, P. Dejax and M. Gendreau (2005a), Traveling salesman problems with profits. *Transportation Science* 39, 188-205.

[13] D. Feillet, P. Dejax and M. Gendreau (2005b), The profitable arc tour problem: solution with a branch-and-price algorithm. *Transportation Science* 39, 539-552.

[14] A. Hertz, G. Laporte and M. Mittaz (2000), A tabu search heuristic for the capacitated arc routing problem. *Operations Research* 48, 129-135.

[15] A. Hertz, G. Laporte and P. Nanchen-Hugo (1999), Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing* 1, 53-62.

[16] S. Lin and B. W. Kernighan (1973), An effective heuristic algorithm for the Traveling Salesman Problem. *Operations Research* 21, 498-516.

[17] V. Maniezzo, T. Stützle, S. Voß (2010), *Matheuristics. Hybridizing Metaheuristics and Mathematical Programming.* Annals of Information Systems, Vol. 10, Springer, New York.

[18] C.E. Miller, A.W. Tucker and R.A. Zemlin (1960), Integer programming formulation of travelling salesman problem., *Journal of ACM* 3, 326-329.

[19] J.R. Montoya-Torres, A.A. Juan, L.H. Huatuco, J. Faulin and G.L. Rodriguez-Verjan, eds. (2012) *Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing, Scheduling and Availability Solutions.* IGI Global.

[20] P. Vansteenwegen, W. Souffriau and D. Van Oudheusden (2011), The orienteering problem: a survey. *European Journal of Operational Research* 209, 110.

[21] E.E. Zachariadis and C.T. Kiranoudis (2011). Local search for the undirected capacitated arc routing problem with profits. *European Journal of Operational Research* 210, 358-367.

				$p = 0$		$p = 0.25$		$p = 0.5$	
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 of classes R, D and G

		$p = 0$			$p = 0.25$			$p = 0.5$		
		R30	R40	R50	R30	R40	R50	R30	R40	R50
	# solved	5	5	5	5	5	5	5	5	5
	# opt	5	5	5	5	5	5	5	5	4
K=2	Av. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.79
	Av. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.79
	Max. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	28.97
	Max. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	28.97
	# solved	5	5	5	5	5	5	5	5	5
	# opt	5	5	5	5	5	5	5	5	5
K=3	Av. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Av. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Max. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Max. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	# solved	5	5	5	5	5	5	5	5	5
	# opt	5	5	5	5	5	5	5	5	5
K=4	Av. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Av. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Max. Gap	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Max. Gap*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table II: Computational results for the sets of instances R30, R40, and R50

		$p = 0$			$p = 0.25$			$p = 0.5$		
		D36	D64	D100	D36	D64	D100	D36	D64	D100
	# solved	9	9	9	9	9	9	9	9	9
K=2	# opt	9	5	0	9	5	1	9	4	1
	Av. Gap	0.00	0.48	2.58	0.00	0.26	2.92	0.00	1.10	4.71
	Av. Gap*	0.00	0.48	2.58	0.00	0.26	2.92	0.00	1.10	4.71
	Max. Gap	0.00	1.79	5.40	0.00	1.20	10.68	0.00	5.12	12.42
	Max. Gap*	0.00	1.79	5.40	0.00	1.20	10.68	0.00	5.12	12.42
	# solved	9	9	4	9	9	5	9	9	7
K=3	# opt	9	7	2	8	6	3	9	5	1
	Av. Gap	0.00	0.10	3.31	0.08	0.42	4.14	0.00	2.05	20.45
	Av. Gap*	0.00	0.10	0.19	0.08	0.42	1.33	0.00	2.05	8.91
	Max. Gap	0.00	0.75	12.50	0.74	1.64	9.34	0.00	5.78	20.07
	Max. Gap*	0.00	0.75	0.62	0.74	1.64	5.31	0.00	5.78	20.07
	# solved	9	5	2	9	7	4	9	7	5
K=4	# opt	9	4	2	9	6	3	9	6	3
	Av. Gap	0.00	1.42	4.58	0.00	0.68	3.02	0.00	1.21	7.53
	Av. Gap*	0.00	0.18	0.00	0.00	0.15	0.05	0.00	0.30	2.53
	Max. Gap	0.00	4.05	10.98	0.00	4.57	9.27	0.00	5.36	22.07
	Max. Gap*	0.00	0.91	0.00	0.00	1.02	0.20	0.00	2.07	11.40

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

		$p = 0$			$p = 0.25$			$p = 0.5$		
		G36	G64	G100	G36	G64	G100	G36	G64	G100
	# solved	9	9	8	9	9	8	9	9	8
	# opt	9	7	3	9	7	4	9	9	5
K=2	Av. Gap	0.00	0.63	1.56	0.00	1.58	1.53	0.00	0.00	2.62
	Av. Gap*	0.00	0.63	1.35	0.00	1.58	1.21	0.00	0.00	2.03
	Max. Gap	0.00	3.03	4.00	0.00	8.33	4.09	0.00	0.00	7.35
	Max. Gap*	0.00	3.03	4.00	0.00	8.33	3.51	0.00	0.00	6.06
	# solved	9	9	5	9	9	6	9	9	8
	# opt	9	8	2	9	9	3	9	7	3
K=3	Av. Gap	0.00	0.25	3.27	0.00	0.00	3.33	0.00	1.18	5.04
	Av. Gap*	0.00	0.25	1.62	0.00	0.00	1.44	0.00	1.18	4.44
	Max. Gap	0.00	2.27	7.69	0.00	0.00	9.11	0.00	5.88	12.20
	Max. Gap*	0.00	2.27	2.82	0.00	0.00	3.45	0.00	5.88	12.20
	# solved	8	3	1	8	9	4	7	9	6
	# opt	8	3	1	8	9	3	7	9	4
K=4	Av. Gap	0.38	4.72	8.53	0.53	0.00	6.38	1.96	0.00	10.87
	Av. Gap*	0.00	0.00	0.00	0.00	0.00	1.06	0.00	0.00	1.52
	Max. Gap	3.45	11.74	13.08	4.76	0.00	17.46	11.76	0.00	44.61
	Max. Gap*	0.00	0.00	0.00	0.00	0.00	4.26	0.00	0.00	5.41

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

		$p = 0$	$p = 0.5$
K=2	# solved	10	9
	# opt	0	1
	Av. Gap	0.81	1.52
	Av. Gap*	0.81	1.46
	Max. Gap	1.68	4.11
	Max. Gap*	1.68	4.11
K=3	# solved	10	3
	# opt	1	1
	Av. Gap	0.78	3.15
	Av. Gap*	0.78	2.57
	Max. Gap	3.13	5.69
	Max. Gap*	3.13	5.69
K=4	# solved	2	2
	# opt	1	0
	Av. Gap	5.03	4.11
	Av. Gap*	0.06	1.13
	Max. Gap	11.11	7.87
	Max. Gap*	0.12	1.59

Table V: Computational results for the class of instances T50

Instance	set	K	p	Gap*	# traversed arcs per route	# required and profitable arcs per route	$\frac{\sum_{r \in R_p(s)} T(r)}{T_{max} * k}$
hertzd28	D100	2	0.25	10.68	53.50	24.00	99.82
hertzr16	R50	2	0.5	28.97	14.50	6.50	99.11
hertzd34	D100	2	0.5	12.42	105.00	59.00	99.97
hertzd29	D100	3	0.5	12.08	32.33	13.33	99.63
hertzd32	D100	3	0.5	20.07	46.00	26.33	99.97
hertzd33	D100	3	0.5	13.05	43.00	22.00	99.75
hertzg33	G100	3	0.5	10.81	52.00	26.67	100.00
hertzg35	G100	3	0.5	12.20	56.00	31.33	100.00
hertzd35	D100	4	0.5	11.40	54.00	29.50	98.91

Table VI: Analysis of instances with large gaps

	Class R	Class D	Class G	Class T50
Av. Gap	0.16	2.26	2.01	2.57
Av. Gap*	0.16	1.05	0.66	1.09
# opt (# solved)	134 (135)	144 (208)	173 (206)	4 (36)
	K=2	K=3	K=4	
Av. Gap	0.96	1.78	2.30	
Av. Gap*	0.90	0.81	0.19	
# opt (# solved)	150 (223)	156 (201)	149 (161)	
	$p = 0$	$p = 0.25$	$p = 0.5$	
Av. Gap	1.40	1.01	2.56	
Av. Gap*	0.35	0.39	1.23	
# opt (# solved)	144 (193)	156 (186)	155 (206)	
	Total Av. Gap		**1.68**	
	Total Av. Gap*		**0.67**	
	Total # opt (total # solved)		**455 (585)**	

Table VII: Average results

Instance	MAT	low tabu tenure	no intensification	no route improvement	no sequence move	$\Theta = 5$ in (11)
hertzd28	10.68	2.86	1.94	10.32	5.39	10.14
hertzr16	28.97	0.00	28.97	0.00	28.97	0.00
hertzd34	12.42	6.81	11.99	7.10	6.90	13.28
hertzd29	12.08	inf.	inf.	inf.	51.98	67.08
hertzd32	20.07	18.13	22.99	14.43	15.03	12.48
hertzd33	13.05	15.50	17.40	19.37	16.55	14.26
hertzg33	10.81	10.81	2.50	13.89	5.13	7.89
hertzg35	12.20	15.00	6.98	12.20	12.20	4.55
hertzd35	11.40	2.28	1.44	1.54	3.35	2.28

Table VIII: Computational study on instances with large gaps