A two-phase solution algorithm for the Flexible Periodic Vehicle Routing Problem

Claudia Archetti
Department of Economics and Management
Università di Brescia
Brescia, Italy
claudia.archetti@unibs.it

Elena Fernández
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya
Barcelona Graduate School of Mathematics
Barcelona, Spain
e.fernandez@upc.edu

Diana L. Huerta-Muñoz [1]
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya
Barcelona, Spain
diana.huerta@upc.edu

Last Update May 7, 2018

[1]Corresponding author

Abstract

The Flexible Periodic Vehicle Routing Problem is the problem of visiting a given set of customers considering a certain periodicity to attend their demands. It is a generalization of the Periodic Vehicle Routing Problem where the fixed schedule constraint is relaxed and the quantity to deliver to each customer at each visit is a decision variable. This flexibility leads to remarkable savings in total costs and this explains the interest in studying the problem and developing effective solution approaches. In this work, an iterative two-phase matheuristic is developed to solve medium and large instances of the problem. Computational tests are made on benchmark instances and on newly generated instances. The results of the matheuristic are compared to optimal solutions, on small-size instances, and to lower bounds on larger instances. Computational results show that good quality solutions are obtained in a reasonable amount of time.

Keywords: Flexible Periodic Vehicle Routing; Matheuristic; Service frequency.

1 Introduction

Several real–world applications deal with periodic delivery operations over a given time horizon. Francis et al. (2008) provide an extensive list of such applications, which include delivery of groceries or collection of waste, to name just a few. Two well-known families of routing problems dealing with periodic deliveries are *Periodic Vehicle Routing Problems* (Campbell and Wilson, 2014, PVRPs) and *Inventory Routing Problems* (Bertazzi and Speranza, 2012, 2013; Coelho et al., 2013, IRPs). In PVRPs, service to customers has to be provided over multiple periods. It is assumed that customers must be served with a certain frequency and, according to a given schedule, they must receive a fixed quantity at each visit. IRPs differ from PVRPs in that they do not fix the service frequency nor the quantity delivered to each customer at each visit. Instead, each customer has a predefined rate of goods consumption per period, and the inventory level is calculated at each time period. The distribution plan has to be such that each customer must be able to satisfy his consumption rate at each time period.

Another vehicle routing problem dealing with periodic demand over a planning horizon is the *Flexible Periodic Vehicle Routing Problem* (Archetti et al., 2017, FPVRP). In the FPVRP, each customer has a total demand that has to be satisfied by the end of the planning horizon. The quantity delivered at each visit should not exceed the customer storage capacity, which is typically lower than the total demand. Thus, multiple visits to each customer must be performed over the planning horizon. The FPVRP can be seen as an extension of the PVRP where no fixed frequency schedule is set. Instead, if the customer capacity corresponds to the ratio between the total demand and the frequency established in the PVRP, then the FPVRP becomes a generalization of the PVRP where each customer must be visited a number of times which is at least equal to the PVRP frequency. Moreover, the quantity delivered at each visit has to be defined. This clearly increases the flexibility with respect to the PVRP settings and may produce cost savings. The FPVRP is also related to the IRP, where no fixed frequency is set for the customers visits and the delivered quantity is a decision variable. However, contrary to the IRP, no inventory level is considered in the FPVRP. Again, this gives the FPVRP a higher flexibility with respect to the IRP which may result in cost savings.

The FPVRP was recently introduced in Archetti et al. (2017), where alternative mathematical programming formulations where presented and it was shown that, theoretically, it can produce arbitrarily large improvements with respect to both the PVRP and the IRP, in terms of the overall routing cost. The computational experiments carried out showed that, in practice, these improvements come at the expense of a considerable increase in the computing time, due to the additional decisions that the FPVRP incorporates with respect to the PVRP: the periods when the customers should be visited as well as the amount of product to be delivered at each visit. In particular, in Archetti et al. (2017) it was only possible to solve instances with up to 20 customers and five time periods within four hours of computing time, using CPLEX with the tightest of the two formulations proposed. Moreover, optimality of the obtained solutions could be proven for 45 out of the 100 considered instances.

While the focus of Archetti et al. (2017) was to introduce the FPVRP and to analyze the savings that it may produce with respect to the PVRP and the IRP, in this paper we propose an effective solution method for the FPVRP. In particular, we present a matheuristic for solving large instances of the FPVRP. This matheuristic consists of two phases: *Distribution Plan (DP)-Generation* and *Improvement*. Broadly speaking, the *DP-Generation* splits the decisions of the FPVRP in two subsets and solves independently the subproblem limited to only one subset of decisions. The first step of the *DP-Generation* finds a distribution pattern for each customer, consisting of a *calendar* with the time periods when customers are visited and quantities that are distributed at each time period of the calendar, ignoring all routing aspects. The calendar and quantities are jointly obtained through the solution of a Mixed-Integer Linear Programming (MILP) formulation. The second step focuses on the routing aspects by finding, for each time period, a set of routes following the distribution pattern produced in the first step. Then, the solution obtained is the input of the *Improvement* phase in which a Tabu Search (Glover and Laguna, 1997) is applied to improve it as much as possible. The algorithm is iterative in the sense that, once the tabu search has terminated, the first phase is called again and the procedure is repeated until a stopping criterion is met. Extensive computational experiments have been run to analyze the contribution and the performance of each element of the heuristic and to assess its overall effectiveness. High quality solutions for instances with up to 100 customers and five periods are given. For the instances with 50 customers or more the solutions produced by our algorithm outperform those produced by CPLEX within four hours of computing time. Moreover, the average gap of the obtained solutions relative to the lower bounds produced by the formulation of Archetti et al. (2017) is 3.83%.

This paper is organized as follows. In Section 2, the definition of the FPVRP and its mathematical formulation are presented. Section 3 contains a detailed description of the proposed matheuristic. Computational experience is reported in Section 4. Finally, some concluding remarks and future work are given in Section 5.

2 The Flexible Periodic Vehicle Routing Problem

In this section we provide the definition of the FPVRP and recall the load-based MILP formulation proposed in Archetti et al. (2017), which was the most effective of the two proposed formulations. Consider a complete and directed graph $G = (N, A)$ with set of nodes $N = \{0\} \cup C$ where 0 denotes the depot and $C = \{1, \dots, n\}$ the set of customers. Let $T = \{1, \dots, H\}$ be a discrete set of time periods. Each customer $i \in C$ has a total demand W_i over T and a storage capacity w_i. The quantity

q_i^t delivered at each visit cannot be greater than w_i and the sum of the quantities q_i^t over T must be equal to W_i. A fleet of vehicles $K = \{1, \dots, m\}$, with a homogeneous capacity Q is available to perform the service. A cost $c_{ij} \geq 0$ is associated with each arc $(i, j) \in A$ and is paid every time a vehicle traverses the arc. The FPVRP is defined as the problem of finding a set of routes that minimizes the total routing costs satisfying customer demands.

The load-based MILP formulation for the FPVRP proposed in Archetti et al. (2017) uses the following sets of decision variables:

- $z_i^t = \begin{cases} 1 & \text{If customer } i \text{ is visited at time period } t, \\ 0 & \text{otherwise.} \end{cases} \quad (i \in C, t \in T)$

- $y_{ij}^t = \begin{cases} 1 & \text{If arc } (i,j) \text{ is traversed at time period } t, \\ 0 & \text{otherwise.} \end{cases} \quad ((i,j) \in A, t \in T)$

- z_0^t: Number of vehicles used at time period $t \in T$.

- l_{ij}^t: Load of the vehicles when traversing arc $(i, j) \in A$, at time period $t \in T$.

- q_i^t: Quantity delivered to customer $i \in C$ at time period $t \in T$.

The formulation is as follows:

$$
\begin{aligned}
\min \quad & \sum_{t\in T}\sum_{(i,j)\in A} c_{ij} y_{ij}^t && && (1)\\
s.t. \quad & q_i^t \le w_i z_i^t && i\in C, t\in T && (2)\\
& \sum_{i\in C} q_i^t \le Q z_0^t && t\in T && (3)\\
& \sum_{j|(i,j)\in A} y_{ij}^t = z_i^t && i\in C, t\in T && (4)\\
& \sum_{j|(i,j)\in A} y_{ij}^t = \sum_{j|(j,i)\in A} y_{ji}^t && i\in N, t\in T && (5)\\
& \sum_{j|(i,j)\in A} l_{ij}^t - \sum_{j|(j,i)\in A} l_{ji}^t = \begin{cases} -q_i^t, i\in C \\ \sum\limits_{i'\in C} q_{i'}^t, i=0 \end{cases} && i\in N, t\in T && (6)\\
& l_{ij}^t \le Q y_{ij}^t && (i,j)\in A, t\in T && (7)\\
& \sum_{j|(0,j)\in A} y_{0j}^t \le |K| && t\in T && (8)\\
& z_0^t = \sum_{i\in C} y_{0i}^t && t\in T && (9)\\
& \sum_{t\in T} q_i^t = W_i && i\in C && (10)\\
& \sum_{t\in T}\sum_{j\in C} l_{j0}^t = 0 && && (11)\\
& y_{i0}^t \le \sum_{r\le i} y_{0r}^t && \forall i\in C, \forall t\in T && (12)\\
& q_i^t \ge 0 && i\in C, t\in T && (13)\\
& z_0^t \in \mathbb{Z} && t\in T && (14)\\
& z_i^t \in \{0,1\} && i\in C, t\in T && (15)\\
& y_{ij}^t \in \{0,1\}, l_{ij}^t \ge 0 && (i,j)\in A, t\in T && (16)
\end{aligned}
$$

The objective function (1) minimizes the routing costs. Constraints (2) impose that the quantity delivered to each customer does not exceed w_i. Constraints (3) establish that the total quantity delivered at time t does not exceed the total capacity of the vehicles used at time t. Constraints (4) state that one arc has to exit from a visited node. Constraints (5) are the flow conservation constraints. Constraints (6) are the load conservation constraints. Constraints (7) impose that the vehicle load does not exceed its capacity and link y and l variables. Constraints (8) ensure that the number of vehicles used is at most $|K|$. Constraints (9) ensure that variables z_0^t take the appropriate value which is equal to the number of arcs leaving the depot at each time period. Constraints (10) impose that the total quantity delivered at each customer at the end of the time horizon is equal to

W_i. Constraints (11) establish that all vehicles return to the depot with an empty load. Constraints (12) are symmetry-breaking constraints. Finally, Constraints (13)–(16) define the domain of the variables.

3 A two-phase solution algorithm for the FPVRP

As shown in Archetti et al. (2017), the FPVRP is a very hard problem and only instances of small size can be solved to optimality. Thus, in this section we propose a heuristic approach that can be used to solve medium to large size instances. In particular, we propose a matheuristic algorithm, which iteratively applies two phases until a stopping criterion is met:

1. *Phase 1 (Distribution Plan (DP) Generation)*: Builds an initial feasible solution by solving independent subproblems limited to only a subset of the FPVRP decisions.
2. *Phase 2 (Tabu Search)*: Improving phase, which applies a Tabu Search (TS) algorithm taking as input the solution produced by Phase 1.

Algorithm 1 shows the general framework of the proposed approach.

Algorithm 1 Two-Phase Matheuristic

```
Output:
    $s^*$ : Best solution found
 1: $\bar{\hat{C}}_i^t = 1 \quad i \in C, t \in T$
 2: $\bar{\lambda}_t = 1 \quad t \in T$
 3: $s^* \leftarrow \emptyset$
 4: BestSolCost = BigNumber
 5: while a stopping condition is not true do
 6:     $s \leftarrow$ DP-Generation($\bar{\hat{C}}_i^t, \bar{\lambda}_t$)
 7:     $\bar{s} \leftarrow$ TS($s$)
 8:     if $f(\bar{s}) <$ BestSolCost then
 9:         BestSolCost $= f(\bar{s})$
10:         $s^* \leftarrow \bar{s}$
11:     end if
12:     Update $\bar{\hat{C}}_i^t$
13: end while
14: return ($s^*$)
```

As it is mentioned above, this approach is composed of two main phases (lines 6 and 7) which are applied sequentially until a stopping criterion is achieved. In the first phase, an initial solution s is obtained by applying the DP-Generation. Then, in the second phase, a TS procedure is applied to s to obtain a new solution $\bar{s}$. If $\bar{s}$ improves the best solution s^*, then s^* is updated. Values of $\bar{\hat{C}}$ and $\bar{\lambda}$ are set as follows. At the beginning of the matheuristic they are set to 1. At the following iterations (line 12), if the solution cost $f(\bar{s})$ is different from the solution of the previous iteration, then the removal savings are computed, i.e., if i is visited in period t then $\hat{C}_i^t = c_{\rho s} - (c_{\rho i} + c_{i\varsigma})$, where ρ and ς denote the predecessor and successor of customer i in its route in period t. If i is not visited at t, $\hat{C}_i^t$ is equal to the cheapest insertion cost of i at time t. On the other hand, if $f(\bar{s})$ remains the same, the values of $\hat{C}$ are generated randomly between [-1,-100]. The rationale for this is to provide different initial solutions to the tabu search performed in Phase 2. The value of coefficients λ remains the same as the ones used in the last call to the `DP-Generation` phase. At the end of the matheuristic, the best solution found is reported as the final solution. We now describe each phase in detail.

3.1 Phase 1: DP-Generation

The first phase aims at building an initial feasible solution to the problem. Note that three main decisions have to be taken when dealing with the FPVRP:

1. *Visiting periods*: The periods at which each customer is visited.
2. *Quantities*: The quantity to deliver to each customer at each visit.
3. *Routing*: Vehicle routes in each period. This means determining the assignment of customers to vehicles and, for each vehicle, the sequence in which customers must be visited.

The DP-Generation phase works in two steps. In the first step, it builds a distribution plan handling the first two decisions, i.e., it determines the visiting periods for each customers and the delivered quantities. The distribution plan is then taken as input to the second step which builds vehicle routes.

In particular, the first step consists in solving a MILP called, from now on, `DP-MILP`, which determines the *calendar* i.e., the visiting periods, and the quantities to be delivered to all customers.

The `DP-MILP` makes use of the following notation:

- $\hat{C}_i^t$: Approximated routing cost for visiting customer i in period t.
- λ_t : A parameter greater than or equal to 1 used to penalize infeasibility at time t.

The formulation of the `DP-MILP` is as follows.

$$
\begin{aligned}
\min \quad & \sum_{t\in T}\sum_{i\in C} \hat{C}_i^t z_i^t + z_0^t Q \lambda_t && && (17)\\
s.t. \quad & q_i^t \leq w_i z_i^t && i \in C, t \in T && (18)\\
& \sum_{i\in C} q_i^t \leq \left\lfloor \frac{Q}{\lambda_t} \right\rfloor z_0^t && t \in T && (19)\\
& \sum_{t\in T} q_i^t = W_i && i \in C && (20)\\
& z_0^t \leq |K| && t \in T && (21)\\
& z_i^t \leq 1 && i \in C, t \in T && (22)\\
& q_i^t \geq 0 && i \in C, t \in T && (23)\\
& z_i^t \in \{0,1\} && i \in C, t \in T && (24)\\
& z_0^t \in \mathbb{Z} && t \in T && (25)
\end{aligned}
$$

Variables $\mathbf{z}$ and $\mathbf{q}$ have the same meaning as described in Section 2. The objective function aims at minimizing the sum of the approximated routing costs and an infeasibility penalty. Constraints (18) establish the maximum deliverable quantity to each customer while (19) are vehicle capacity constraints. In particular, constraints (19) are aggregated vehicle capacity constraints that fix the maximum quantity that can be delivered in each time period. This maximum amount corresponds to $\left\lfloor \frac{Q}{\lambda_t} \right\rfloor$ multiplied by the number of vehicles used, with $\lambda_t \geq 1$. Note that a solution satisfying constraints (19) may not be a feasible FPVRP because it may not exist a feasible way to pack quantities q_i^t into z_0^t vehicles. The total demand of each customer is satisfied through constraints (20). Constraints (21) fix the maximum number of vehicles used to $|K|$ while split deliveries are forbidden through (22). Constraints (23)–(25) define the variable domain.

The solution of `DP-MILP` determines, for each time period, which is the subset of visited customers and what is the amount delivered to each of them. This information provides the distribution plan. The DP is then taken as input to the second step which aims at building vehicle routes. In particular, the second step consists in solving a Capacitated Vehicle Routing Problem (Ralph et al., 2003, CVRP) for each time period on the basis of the information provided by the DP. We solve each CVRP through the VRPH package of the Coin OR library (Groër et al., 2010). We use the Clarke and Wright (1964) heuristic implemented in the VRPH package. Note that this algorithm works for the case where there is no limit for the fleet size. Thus, we may obtain a solution where the number of vehicles used in a given period is higher than $|K|$. In this case, another iteration is made by updating the values of $\hat{\mathbf{C}}$ and λ and solving the DP-MILP again. Finally, if after a certain number of iterations ~~(set to 5 in our tests)~~ the solution is still infeasible, we apply a procedure to recover feasibility in which customers of the surplus routes are reallocated in different periods.

The scheme of the phase 1 is sketched in Algorithm 2 where:

- `DP-MILP`$(\hat{\mathbf{C}}, \lambda)$ is the optimal solution of `DP-MILP` when the values of the approximated routing costs and the infeasibility penalties are specified by $\hat{\mathbf{C}}$ and λ, respectively.
- `Routing`(DP) is the solution of the CVRP for each time period on the basis of the distribution plan DP obtained through the VRPH package (Groër et al. (2010)).
- `RecoverFeasibility`(s) is the solution obtained when transforming an infeasible solution into a feasible one. The procedure works as follows.
  - Take one of the periods in which the number of routes is more than m.
  - Select the route with less customers and sequentially remove customers by redistributing the quantity they received in other periods.
  - If all customers of the selected route are reallocated, remove the empty route and follow the same procedure until the number of vehicles used is at most m for all periods.
- `LK`(s) returns an improved solution by applying the Lin-Kernighan algorithm to each route of solution s (Lin and Kernighan, 1973; Helsgaun, 2000). The implementation code for this routine is provided in `http://www.akira.ruc.dk/~keld/research/LKH/`.
- s is the solution obtained at the end of the second step.

Algorithm 2 `DP-Generation`$(\bar{\hat{C}}_i^t, \bar{\lambda}_t)$

```
1:  $\hat{C}_i^t = \bar{\hat{C}}_i^t \quad i \in C, t \in T$
2:  $\lambda_t = \bar{\lambda}_t \quad t \in T$
3:  while a stopping condition is not true do
4:      $DP \leftarrow$ DP-MILP$(\hat{\mathbf{C}}, \lambda)$
5:      $s \leftarrow$ Routing$(DP)$
6:      if $s$ is a feasible FPVRP solution then
7:          Stop and return LK$(s)$
8:      else
9:          Update the values of $\hat{\mathbf{C}}$ and $\lambda$
10:         Go to line 4
11:     end if
12: end while
13: if $s$ is not feasible then
14:     RecoverFeasibility$(s)$
15:     return LK$(s)$
16: end if
```

As shown in Algorithm 2, the values of $\hat{\mathbf{C}}$ and λ are initialized to $\bar{\hat{C}}$ and $\bar{\lambda}$, respectively (line 1). Then, if an infeasible solution is obtained, they are updated (line 9) as follows:

a) $\hat{\mathbf{C}}$: if customer i is visited at time t, then $\hat{C}_i^t$ is equal to the removal savings $\hat{C}_i^t = c_{\rho i} + c_{i\varsigma} - c_{\rho\varsigma}$, where ρ and ς denote the predecessor and successor of customer i in its route of period t. Instead, if customer i is not visited at time t, then $\hat{C}_i^t$ is equal to the cheapest insertion cost of i at time t.

b) λ: if at t the number of vehicles is not greater than $|K|$, then the value of λ_t remains unchanged, otherwise it is increased by ϵ, i.e., $\lambda_t = \lambda_t + \epsilon$. If the DP-MILP becomes infeasible (because of a too large value of λ_t), then, $\lambda_t = \max\{1, \lambda_t - \epsilon\}$.

Note that, once a feasible solution s is obtained, the LK algorithm is applied to each route in s in an attempt to reduce the routing cost (lines 7 and 15).

3.2 Phase 2: Tabu Search heuristic

Phase 2 aims at improving the solution obtained at the end of the first phase. The idea is to define different neighborhoods and embed them in a Tabu Search scheme (Glover and Laguna, 1997) where each selected move is recorded and considered *tabu* for a certain number of iterations. Each neighborhood is explored exhaustively, unless a selected move improves the incumbent. In that case the exploration is stopped and the improving solution is chosen as the next solution. In all other cases, the best non–tabu move in the neighborhood is kept and the best move among all neighborhoods is performed and considered *tabu* for a certain number of iterations. Let $\mathcal{N} = \{N_1, ..., N_l\}$ be the set of neighborhoods with $|\mathcal{N}| = l$. The scheme of the TS is provided in Algorithm 3.

Algorithm 3 TS(s)

```
Input:
    $s$ : Initial Solution
Output:
    $s^*$ : Best solution found
 1: IterNImp, $Iter$, $l$;
 2: BestSolCost $= f(s)$
 3: $s \leftarrow$ SplitOperator(s)
 4: while ($iter <$ IterNImp) do
 5:     BestLocalCost = BigNumber
 6:     Tenure = computeTenure()
 7:     $\tilde{l} = 1$
 8:     repeat
 9:         $s' \leftarrow$ Explore Neighborhood $N_{\tilde{l}}(s)$
10:         if $f(s') <$ BestLocalCost then
11:             BestLocalCost=$f(s')$
12:             $\tilde{s} \leftarrow s'$
13:         end if
14:         $\tilde{l} \leftarrow \tilde{l} + 1$
15:     until $\tilde{l} \leq l$
16:     Update tabu list TL($\tilde{s}$) = iter + Tenure
17:     $\tilde{s} \leftarrow$ LK($\tilde{s}$)
18:     $iter = iter + 1$
19:     if $f(\tilde{s}) <$ BestSolCost then
20:         BestSolCost $= f(\tilde{s})$
21:         $s^* \leftarrow \tilde{s}$
22:         $iter = 0$
23:     end if
24: end while
25: return $s^*$
```

The TS begins with an initial feasible solution obtained in Phase 1. The routes of this solution are split (if possible) using the `SplitOperator` in order to increase the possibility of applying moves which may improve the solution during the search. The resulting solution is the initial solution for all neighborhoods in $\mathcal{N}$. The neighborhoods are explored independently and the best solution among all of them is selected. The corresponding move is considered *tabu* for a certain number of iterations (line 6). Then, the Lin-Kernighan algorithm is applied to the best solution found (line 17). This procedure stops when a maximum number of iterations without improvement is reached (line 4).

We now describe in detail the three main ingredients of the proposed TS: the `SplitOperator`, the set of neighborhoods and the tabu lists.

A) The `SplitOperator`: This operator aims at splitting one route into two routes without increasing the solution cost. This situation happens when a route travels through an edge (i, j) whose cost c_{ij} is equal to $c_{i0} + c_{0j}$. In this case, the route is split in two smaller routes traversing edges $(i, 0)$ (first route) and $(0, j)$ (second route). This is done only if the total number of routes used is lower than mH. The idea behind the `SplitOperator` is to create routes with a larger residual capacity and, thus, to allow a wide range of modifications when applying the neighborhoods described in the following. When a route at time t is split, two situations may happen:

- **The number of routes used at time t is lower than m.** In this case, the two new routes are both performed at time t and no further change is made.
- **The number of routes used at time t is m.** In this case, we cannot assign both new routes to time t. Thus, at least one route must be performed in a different day. Let us define:
  - r_1 and r_2: the two routes obtained from the splitting.
  - S_r: subset of customers served in route r.
  - $S(t)$: subset of customers served at time t.
  - $\tilde{q}_r$: residual capacity of route r.
  - r_i^t: route serving customer i at time t.

  Then, a route r can be moved from time t to time $t' \neq t$ if $S_r \cap S(t') = \emptyset$ or the following holds for each customer $i \in S_r \cup S(t')$:

  1. $q_i^t + q_i^{t'} \leq w_i$ and
  2. either $q_i^t \leq \tilde{q}_{r_i^{t'}}$ or $q_i^{t'} \leq \tilde{q}_r$.

  Thus, if either r_1 or r_2 satisfies the above conditions the split is performed, otherwise it is discarded.

B) Neighborhoods: They are listed in the order they are applied according to Algorithm 3.

1. *Intra-period moves*: The following moves are applied to each period independently. Thus, they do not have an impact on the subset of customers visited in each period, which remains unchanged.
   (a) *1-move* (N_1): Consider a customer visited in period t. Remove the customer from the route that currently serves it and insert it in another route, using the cheapest insertion rule. The best route (in terms of insertion cost) that can feasibly accommodate the quantity delivered to the customer is chosen.
   (b) *1-swap* (N_2): Consider two customers i and j served in two different routes, r and r', in period t and swap the two customers. The swap is made as follows. We first remove both customers from their current route and then insert them in the new route through the cheapest insertion method, as done in the *1-move*.

2. *Inter-period moves*: These moves are applied to pairs of periods t and t' in order to change the visit plan of customers. Similar to the intra-period moves, there are two neighborhoods considered.
   (a) *1-move* (N_3): Consider a customer visited in period t. Remove the customer from the route that currently serves it and insert it in a route in period t', using the cheapest insertion rule as done in the *1-move* intra-period.
   (b) *1-swap* (N_4): Consider customer i served in t and customer j served in t' and swap them. The swap is made as follows. We first remove both customers from their current route. The insertion is made by applying the cheapest insertion method to all routes performed in the period where the customer has to be inserted and choosing the best one.

In any of the above mentioned neighborhoods, every time a customer is removed from a route and inserted in another one, the same quantity delivered in the original route is moved to the newly assigned route, if this is feasible. If this is not feasible (either because the vehicle capacity or the customer capacity are exceeded), then the excess quantity is assigned to the other customer visits if feasible, i.e., if neither vehicle capacity nor customer capacity are exceeded and the move made in the corresponding period is not tabu. The assignment is done in chronological order, i.e., from the first to the last visit. If the excess quantity cannot be reassigned, then the move is infeasible and, thus, discarded.

For each neighborhood, all feasible non-tabu moves are evaluated and the best one is chosen unless there is a move which improves the incumbent (aspiration criterion). In that case, the evaluation stops (for all neighborhoods) and the solution obtained with that move is chosen as the best one. That is, the algorithm exits from the loop in lines 8–15 in Algorithm 3 and goes to line 17. If this is not the case, once all the neighborhoods are explored, the best non-tabu solution is chosen and it becomes tabu for a certain number of iterations as detailed in Algorithm 3.

C) Tabu lists: There are two different tabu lists, one for intra–period moves and another one for inter–period moves:

- Intra–period list $\texttt{TL}(i, r, t)$: If customer i is removed from route r in period t, then it is tabu to reinsert i in r in period t for a certain number of iterations.
- Inter–period list $\texttt{TL}(i, t)$: If customer i is removed from period t, then it is tabu to reinsert i in period t for a certain number of iterations.

The number of iterations for which a move remains tabu is determined by the tabu *tenure*, i.e., a performed move cannot be repeated unless a specific number of iterations (or *tenure*) is met. Given that this value must be selected carefully to avoid the cycling of the TS, we have evaluated the following alternatives:

i) $\texttt{Tenure1} = 10$
ii) $\texttt{Tenure2} = 5 + \lfloor \text{randN}(1, \alpha \cdot \sqrt{|C|R(s')}) \rfloor$
iii) $\texttt{Tenure3} = \lfloor \alpha \cdot (|N| + \text{randN}(1, \sqrt{|C|R(s')})) \rfloor$
iv) $\texttt{Tenure4} = \lfloor \text{randN}(1, \alpha \cdot \texttt{IterNImp}) \rfloor$

where randN(a,b) is a number in [a,b]. Option *i)* corresponds to a constant tenure. Alternative *ii)* corresponds to the tenure proposed by Archetti et al. (2012), which considers a fixed value plus a random term from an interval that depends on the number of customers and the number of routes of the current solution. Alternative *iii)* is similar to the previous option but the fixed term now depends on the number of nodes of the evaluated instance. Finally, option *iv)* assigns a random tenure based on the number of iterations without improvement of the TS.

4 Computational Experiments

In this section we describe the computational experiments that we have conducted and present the numerical results obtained. The matheuristic was implemented in C++ and the `DP-MILP` was solved using the ILOG Concert Technology API (CPLEX 12.5.0.0). All tests were carried out on a HP Intel(R)-Xeon(R) 2.4GHz Workstation with 32GB RAM (Win Server 2012, 64 bits).

Before presenting the numerical results, we describe the sets of benchmark instances used for the tests. Data instances and complementary information can be found at `http://or-brescia.unibs.it/instances`.

4.1 Benchmark instances

Tests have been performed on two sets of instances: the *Training set* and the *Testing set*. Instances of the training set have been used to tune the parameters of the algorithm. The algorithm performance have then been evaluated on instances of the testing set. We now describe how we generated the two sets of instances.

4.1.1 Training set

This set considers a total of 30 instances with $n = 10, 30$, and 50 customers (10 for each size), the depot is located at point (0.5,0.5), and the geographical distribution and the storage capacity of customers are generated uniformly in an interval $X \sim U[0,1]$ and $Y \sim U[0,1]$ (for customer location) and $w_i \sim U[1,Q]$ (for storage capacity), where $Q = 300$. The frequency of visits to customers is set to $f_i = 2, 3, 5$ in a random form and the number of vehicles is computed as $m = 1 + \frac{\sum_{i \in C} w_i f_i}{QH}$, where $H = 5$.

4.1.2 Testing set

This set of instances is classified into three different subsets of instances:

- **Set 1 (S1)**: A set of 5 PVRP benchmark instances from the literature (see, for instance, Chao et al. (1995); Cordeau et al. (1997); Baldacci et al. (2011)) and available at `http://neumann.hec.ca/chairedistributique/data/pvrp/old/`.

- **Set 2 (S2)**: A set of 35 FPVRP instances with *clustered* customers similar to the ones generated in Archetti et al. (2017). These instances are characterized by a value r which is the radius used to generate clusters of customers. All instances of this set have a time horizon $H = 5$. There are 5 instances for each combination of $|C| \in \{10, 15, 20\}$ and $r \in \{0.15, 0.30\}$, plus 5 more instances with $|C| = 20$, $r = 0.50$ and vehicles capacities $Q = 200$, 250 and 300 for 10, 15 and 20 customers, respectively. These instances have been slightly modified from the ones in Archetti et al. (2017). In particular, the number of available vehicles is now computed as $m = 1 + \frac{\sum_{i \in C} w_i f_i}{QH}$ while in Archetti et al. (2017) it is set to $m = 1 + \frac{\sum_{i \in C} w_i}{Q}$. The reason for this change is that the number of vehicles used in the optimal solutions of Archetti et al. (2017) was much smaller than the number of available vehicles. All other data remained unchanged.

- **Set 3 (S3)**: A new set of 10 larger instances, generated in a similar way to the ones in S2, with $|C| \in \{50, 100\}$, $r \in \{0.15\}$ and vehicle capacity of $Q = 500$. Like in the instances of S2, the number of vehicles is computed as $m = 1 + \frac{\sum_{i \in C} w_i f_i}{QH}$.

Detailed information about instances of the testing set is given in Table 1.

Set	r	**Instance**	$\|N\|$	T	f_i	Q	$\|K\|$	$\sum_{i\in C} W_i$
S1		p01	51	2	1	160	2	937
		p14	21	4	1,2,4	20	2	120
		p15	39	4	1,2,4	30	2	200
		p16	57	4	1,2,4	40	2	280
		p32	154	6	2,3,5	20	9	1134
S2	**0.15**	n10k5t5_1	11	5	2,3,5	200	5	3760
		n10k4t5_2					4	2640
		n10k5t5_3					4	2869
		n10k4t5_4					4	2363
		n10k8t5_5					7	5107
		n15k10t5_1	16	5	2,3,5	250	9	9015
		n15k6t5_2					5	4684
		n15k10t5_3					7	7202
		n15k8t5_4					7	7175
		n15k7t5_5					6	5526
		n20k10t5_1	21	5	2,3,5	300	7	8842
		n20k12t5_2					9	11334
		n20k11t5_3					9	10957
		n20k10t5_4					9	11102
		n20k10t5_5					7	8976
	0.30	n10k6t5_1	11	5	2,3,5	200	5	3562
		n10k6t5_2					5	3922
		n10k5t5_3					5	3114
		n10k5t5_4					5	3038
		n10k8t5_5					7	5139
		n15k9t5_1	16	5	2,3,5	250	7	6877
		n15k9t5_2					8	7642
		n15k7t5_3					6	5973
		n15k7t5_4					5	4621
		n15k6t5_5					6	4947
		n20k10t5_1	21	5	2,3,5	300	8	9551
		n20k12t5_2					9	11675
		n20k10t5_3					7	8047
		n20k13t5_4					10	12723
		n20k12t5_5					9	11996
	0.50	n20k14t5_1	21	5	2,3,5	300	10	13225
		n20k10t5_2					7	8862
		n20k7t5_3					6	7490
		n20k10t5_4					7	8373
		n20k11t5_5					9	11437
S3	**0.15**	n50k10t5_1	51	5	2,3,5	500	10	22226
		n50k8t5_2					8	15690
		n50k9t5_3					9	19145
		n50k9t5_4					9	19737
		n50k11t5_5					11	23282
		n100k17t5_1	101	5	2,3,5	500	17	38700
		n100k18t5_2					18	40870
		n100k20t5_3					20	46018
		n100k21t5_4					21	48959
		n100k18t5_5					18	40782

Table 1: Instance information: Testing set.

4.2 Calibration of parameters

We now describe the tests we run in order to obtain the best setting of parameters for the proposed two-phase algorithm. The set of instances used for the computational experiment of this section corresponds to the *Training set (TrS)*. The response variable used to determine the best parameter values is the Average Relative Percentage Deviation (ARPD):

$$\text{ARPD} = \frac{1}{P} \sum_{s \in \text{TrS}} \left(\frac{1}{R} \sum_{r=1}^{R} \frac{f(s,r) - f^{best}(s)}{f^{best}(s)} \right) \times 100 \tag{26}$$

where $f(s,r)$ is the objective function value obtained in the r-th run with instance s of the Training set and $f^{best}(s)$ is the best objective function value found under all the specified tested conditions. The value of R corresponds to the number of runs of the two-phase algorithm performed for each instance, and P is the number of tested instances.

According to preliminary tests, neighborhood N_4 was the most resource-consuming neighborhood for the TS, providing a small average frequency of use but a high computation time requirement on average. Removing N_4 from the TS reduces significantly the computing time and the average solution quality is not affected. Therefore, all subsequent tests were carried out with TS without N_4.

The tuning of the parameters has been performed in two steps. First, the parameter ϵ, corresponding to the DP-Generation phase, is evaluated. Second, a statistical analysis has been applied to evaluate the impact of the interaction among parameters related to the TS phase.

4.2.1 Calibration of Phase 1: ϵ

We made a first test to set the value of the ϵ parameter used in the DP-Generation phase, as specified in Section 3.2, to control feasibility during the construction of the initial solution. We fixed the maximum number of iterations to $\texttt{IterDP}_{\texttt{max}} = 7$. Three values of ϵ are considered: 0.05, 0.10, and 0.15, and only the DP-Generation phase is tested.

Figure 1 depicts the ARDP values for the different values of ϵ tested. As can be seen, in general, the DP-Generation phase obtained better ARPDs values when $\epsilon = 0.10$ in comparison to $\epsilon = 0.05, 0.15$.

Thus, we have fixed $\epsilon = 0.10$ for the following tests.

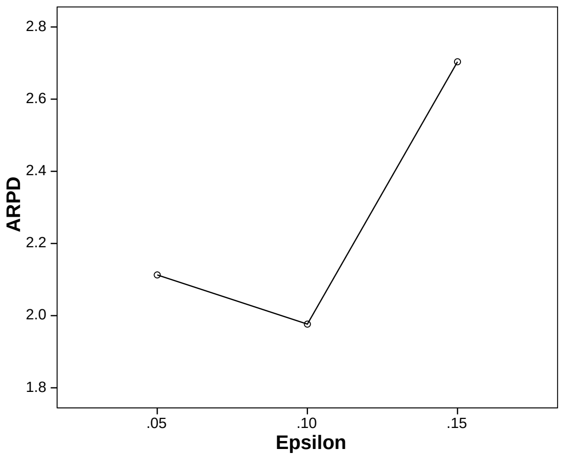


Figure 1: ϵ evaluation.

4.2.2 Calibration Phase 2: Tenure, α, and IterNImp

A second set of tests has been performed to analyze the impact of the different tenure options considered. For this we have used three different values of α_t for the tenure, and three different values of IterNImp, the maximum number of iterations without improvement, for the stopping criterion for the TS.

As indicated in Section 3.2, the following options of `Tenure` are considered:

- $\texttt{Tenure1} = 10$
- $\texttt{Tenure2} = 5 + \lfloor \text{randN}(1, \alpha \cdot \sqrt{|C|R(s')}) \rfloor$
- $\texttt{Tenure3} = \lfloor \alpha \cdot (|N| + \text{randN}(1, \sqrt{|C|R(s')})) \rfloor$
- $\texttt{Tenure4} = \lfloor \text{randN}(1, \alpha \cdot \texttt{IterNImp}) \rfloor$

Except for Tenure1, where the tenure value is constant, the tested values of α are 0.25, 0.50, and 0.75, and the tested values values of IterNImp are 10n, 15n, and 20n, were n is the instance size. For this analysis we have not considered the 50 customer instances, because in all cases the time limit was reached before completing the experiment. Given that a criterion of randomness is involved in most of the `Tenure` options and in the update procedure of the objective function at each iteration of the matheuristic, five runs of the algorithm, for each instance, have been tested. In total, 3400 runs were carried out for the analysis (20 instances $\times$ 4 types of tenure $\times$ 3 values of α $\times$ 3 IterNImp options $\times$ 5 runs).

Table 2 shows the results obtained in the tests. Each cell represents the ARPD among all instances of a specific size and combination of factors. These results show that, in general, the best average results are obtained when `Tenure3` applies a value of $\alpha = 0.75$ (0.26 and 1.14, respectively).

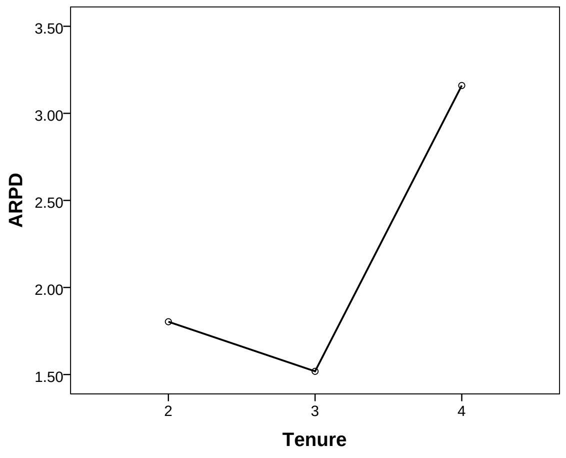
			α		
n	**Tenure**	**IterNImp**	**0.25**	**0.50**	**0.75**
10	`Tenure1`	10n	0.55	0.55	0.55
		15n	0.59	0.59	0.59
		20n	0.56	0.56	0.56
		Avg	**0.56**	**0.56**	**0.56**
	`Tenure2`	10n	1.01	0.46	0.50
		15n	0.82	0.39	0.30
		20n	0.67	0.19	0.38
		Avg	**0.83**	**0.34**	**0.39**
	`Tenure3`	10n	1.82	0.86	0.29
		15n	1.55	0.44	0.25
		20n	1.60	0.19	0.24
		Avg	**1.66**	**0.50**	**0.26**
	`Tenure4`	10n	2.71	1.94	1.39
		15n	2.44	1.32	0.72
		20n	1.60	0.81	0.20
		Avg	**2.25**	**1.36**	**0.77**

			α		
n	**Tenure**	**IterNImp**	**0.25**	**0.50**	**0.75**
30	`Tenure1`	10n	3.52	3.52	3.52
		15n	3.54	3.54	3.54
		20n	3.72	3.72	3.72
		Avg	**3.59**	**3.59**	**3.59**
	`Tenure2`	10n	4.11	3.12	2.61
		15n	3.66	2.65	2.21
		20n	3.73	3.37	2.30
		Avg	**3.83**	**3.05**	**2.37**
	`Tenure3`	10n	3.88	2.36	1.22
		15n	3.26	1.97	0.95
		20n	3.25	1.95	1.25
		Avg	**3.46**	**2.09**	**1.14**
	`Tenure4`	10n	6.01	4.92	4.61
		15n	5.78	4.67	4.08
		20n	4.88	4.72	4.06
		Avg	**5.56**	**4.77**	**4.25**

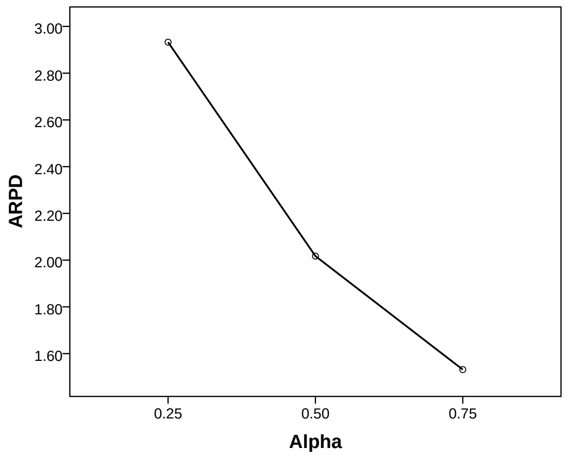
Table 2: ARPD for each combination of Tenure, α, and IterNImp.

A non-parametric Kruskal-Wallis test has been carried out with the obtained results, where the ARPD has been used as the response variable. We excluded the results for `Tenure1` as it does not consider the α parameter. The analysis indicates that, as it was expected, the parameters `Tenure` and α determine the greatest impact in the final ARPD.

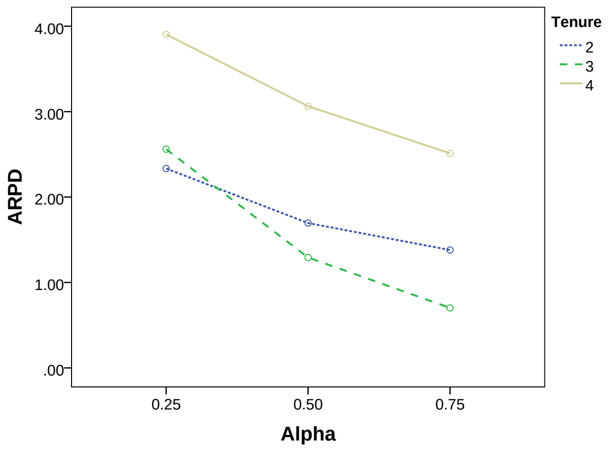
Figures 2a-2b compare the ARPDs for the different `Tenure` and α options, respectively. We can observe that, in general, `Tenure3` is the option that outperforms the remaining alternatives. The same is true for $\alpha = 0.75$, which outperforms the other tested values of α. On the other hand, Figure 2c shows the interaction between α and `Tenure`. The combination that produces smaller ARPDs is the corresponding to `Tenure3` with $\alpha = 0.75$. Figure 2d shows that when Tenure3 is used to set the tenure value, the best results are obtained with IterNImp=15n and IterNImp=20n, for all values of α. Thus, we fixed IterNImp=15n.



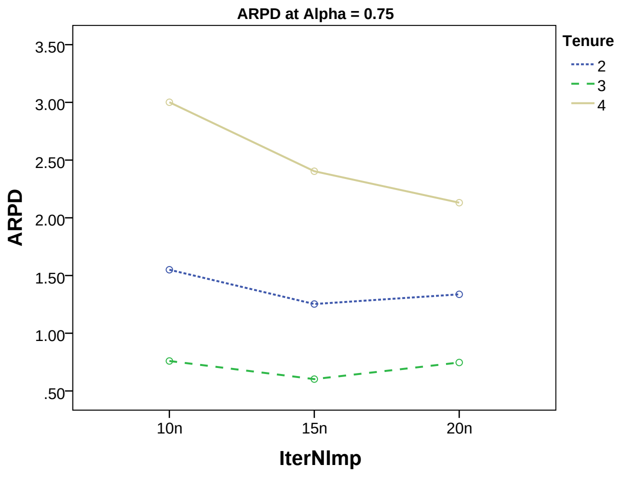
(a) General ARPD for each Tenure option.



(b) General ARPD for each α option.



(c) Tenure and α interaction.



(d) IterNImp and α interaction.

Figure 2: Graphical representation of the interaction among parameters of Phase 2.

Summarizing, we set `Tenure3` with a value of $\alpha = 0.75$, and a stopping criterion of `IterNImp`=15n for the proposed TS in the following tests.

4.3 Heuristic performance

The final computational tests were carried out with the calibrated two-phase algorithm, using the complete *Testing set* of benchmark instances. The main objective of these tests is to assess the effectiveness of the two-phase algorithm and to compare its results with the results given by the FPVRP formulation. We used the best setting of parameter obtained in the previous section, which is summarized in Table 3.

Matheuristic phases	**Parameters**
Overall	Time limit = 14400 Global iterations = 10 Runs = 5 ($n < 50$), 1 ($n \geq 50$)
Phase 1	$\texttt{IterDP}_{\texttt{max}} = 7$ $\epsilon = 0.10$
Phase 2	`IterNImp` = 15n `Tenure` = $\lfloor \alpha \cdot (\lvert N \rvert + \text{randN}(1, \sqrt{\lvert C \rvert R(s')})) \rfloor$ $\alpha = 0.75$

Table 3: Parameter setting information.

The two-phase algorithm, is repeated for 10 iterations, provided that it does not exceed a maximum computing time of 14400 seconds (the same time limit used to obtain the best-known solution). That is, the algorithm stops when one of the two conditions is met. $\texttt{IterDP}_{\texttt{max}}$ determines the maximum number of iterations of Phase 1 (lines 3–12 of Algorithm 2). This limit will not be reached if the current `DP-MILP` produces a feasible solution.

As the two-phase algorithm includes some randomness, we tested each instance five times when the number of customers is lower than 50. For instances with 50 or more customers, we tested each instance only once as each run required four hours of computation.

Table 4 summarizes the obtained results and compares them with those obtained with CPLEX with the FPVRP formulation. In this table, column **BestKnown** shows the value of the best-known solution for each instance, columns in block **FPVRP** reproduce the results of Archetti et al. (2017) obtained with the FPVRP formulation (best solution, best lower bound, and computing time), columns in **Math-FPVRP** give information related to the performance of the two-phase algorithm (best solution found, average solution among all runs, and the required average and minimum/-maximum computing times). On the other hand, columns of block **Gap**, give the percentage gaps of the best solution produced by the heuristic with respect to best-known solutions (column **BK**) and with respect to the best lower bound produced by the FPVRP formulation (column **LB**). The entries of column **BK** have been computed with the expression $\frac{Z_{\text{Heur}} - Z_{\text{BEST}}}{Z_{\text{Heur}}} \times 100\%$. Note that negative entries in this column indicate that the heuristic results improve the solutions obtained with the formulation. Percentage deviations in column **LB** have been computed with the expression $\frac{Z_{\text{Heur}} - Z_{\text{BestLB}}}{Z_{\text{BestLB}}} \times 100\%$.

Average BK gaps range between -8.74% and 2.52%, with a total average gap of -0.30%. No BK gaps are reported for the largest instances since CPLEX is not able to obtain any feasible solution within the allowed computing time. Percentage deviations relative to the lower bounds, given in column LB, range between 2.22% and 8.07%, with a total average gap of 3.83%. In general, these

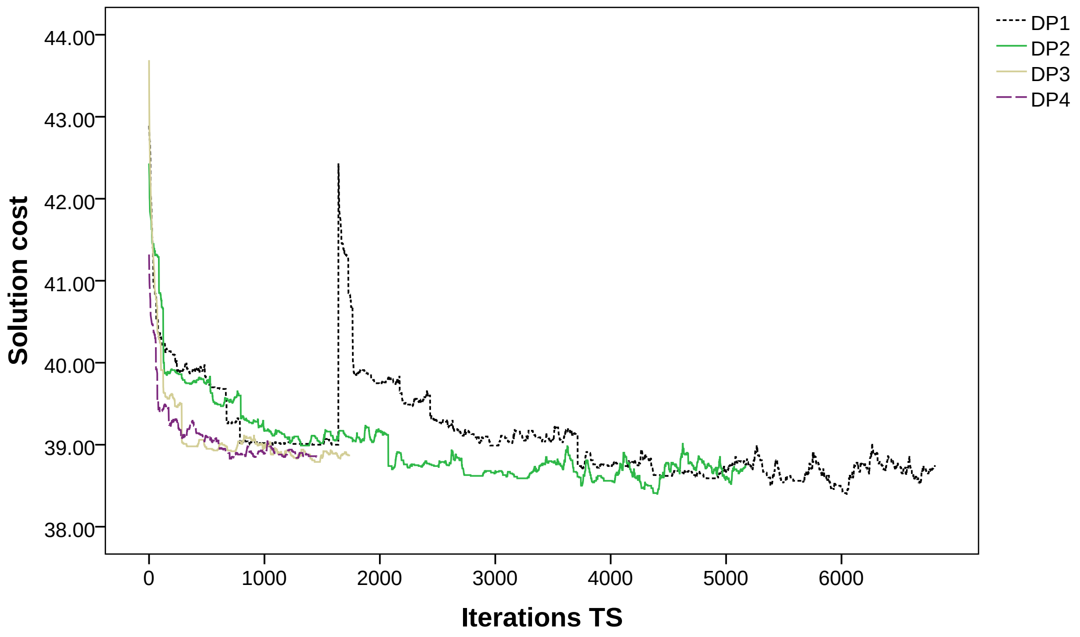
				FPVRP			**Math-FPVRP**				**Gap**	
Instance	r	$\lvert N\rvert$	**BestKnown**	**BestSol**	**BestLB**	**Time**	**BestSol**	**AvgSol**	$\mathbf{T_{avg}}$	$\mathbf{T_{min}/T_{max}}$	**BK**	**LB**
p01		51	**524.61**	524.93	510.46	14399	536.83	536.83	3422.43	3422.43	2.28%	5.17%
p14		20	**954.81**	954.81	954.71	11525	954.84	954.84	932.46	766.73 / 1062.64	0.00%	0.01%
p15		28	**1862.63**	1862.63	1825.04	14399	1862.68	1862.68	3261.75	2339.06 / 4081.59	0.00%	2.06%
p16		56	**2875.24**	2875.24	2814.29	14399	2875.28	2875.28	10295.90	10295.90	0.00%	2.17%
p32		154	**78072.88**	—	40990.75	14400	87066.60	87066.60	14489.90	14489.90	10.33%	112.41%[1]
										Avg.	**2.52%**	**2.35%**
n10k5t5_1			**20.79**	20.79	20.27	14400	20.82	21.20	73.90	68.69 / 81.69	0.14%	2.71%
n10k4t5_2			**12.44**	12.44	12.35	14400	12.43	12.43	73.32	67.06 / 81.84	-0.08%	0.65%
n10k5t5_3		11	**13.23**	13.23	12.91	14400	13.47	13.47	313.54	243.59 / 362.22	1.78%	4.34%
n10k4t5_4			**13.53**	13.53	13.05	14400	13.47	13.63	148.92	105.40 / 239.40	-0.45%	3.22%
n10k8t5_5			**25.91**	25.91	25.58	14400	27.21	27.34	74.06	65.46 / 83.58	4.78%	6.37%
										Avg.	**1.24%**	**3.46%**
n15k10t5_1			**34.28**	34.28	33.48	14400	34.71	34.84	285.70	254.66 / 307.53	1.24%	3.67%
n15k6t5_2			**17.41**	17.41	16.54	14400	17.41	17.41	867.56	533.58 / 1075.90	0.00%	5.26%
n15k10t5_3	0.15	16	**25.24**	25.24	24.42	14400	24.77	25.05	451.71	384.37 / 544.93	-1.90%	1.43%
n15k8t5_4			**32.12**	32.12	30.97	14400	32.30	32.71	560.71	374.62 / 762.17	0.56%	4.29%
n15k7t5_5			**23.92**	23.92	22.87	14400	23.91	23.95	505.15	373.87 / 678.35	-0.04%	4.55%
										Avg.	**-0.03%**	**3.84%**
n20k10t5_1			**24.58**	24.58	23.57	14400	24.39	24.67	936.85	837.61 / 1000.83	-0.78%	3.48%
n20k12t5_2			**36.08**	36.08	35.49	14400	36.29	36.63	1399.24	1215.30 / 1610.31	0.58%	2.25%
n20k11t5_3		21	**23.69**	23.69	22.76	14400	23.75	24.13	1065.96	870.10 / 1338.27	0.25%	4.35%
n20k10t5_4			**35.36**	35.36	34.60	14400	35.50	35.59	914.73	863.33 / 1004.97	0.39%	2.60%
n20k10t5_5			**29.44**	29.44	28.88	14400	29.91	30.02	1096.23	961.87 / 1147.90	1.57%	3.57%
										Avg.	**0.40%**	**3.25%**
n10k6t5_1			**19.04**	19.04	18.99	14400	19.33	19.34	105.16	92.95 / 112.68	1.50%	1.79%
n10k6t5_2			**13.89**	13.89	13.63	14400	13.88	13.88	111.47	105.80 / 116.98	-0.07%	1.83%
n10k5t5_3		11	**14.50**	14.50	14.10	14400	14.65	14.65	79.51	66.41 / 92.17	1.02%	3.90%
n10k5t5_4			**14.39**	14.39	14.11	14400	14.40	14.42	188.65	170.32 / 214.98	0.07%	2.06%
n10k8t5_5			**19.39**	19.39	19.33	14400	19.82	19.84	87.50	77.51 / 93.55	2.17%	2.53%
										Avg.	**0.94%**	**2.42%**
n15k9t5_1			**27.16**	27.16	26.65	14400	27.14	27.14	361.97	298.17 / 455.51	-0.07%	1.84%
n15k9t5_2			**29.72**	29.72	29.20	14400	30.09	30.37	361.50	304.67 / 414.82	1.23%	3.05%
n15k7t5_3	0.30	16	**27.84**	27.84	27.38	14400	28.08	28.19	404.67	358.06 / 486.18	0.85%	2.56%
n15k7t5_4			**17.42**	17.42	17.07	14400	17.43	17.43	494.76	393.35 / 547.97	0.06%	2.11%
n15k6t5_5			**20.58**	20.58	20.22	14400	20.53	20.59	326.78	282.33 / 377.11	-0.24%	1.53%
										Avg.	**0.36%**	**2.22%**
n20k10t5_1			**29.59**	29.59	29.06	14400	30.06	30.32	1136.03	1048.01 / 1350.10	1.56%	3.44%
n20k12t5_2			**31.50**	31.50	30.82	14400	31.85	32.03	812.81	692.63 / 952.91	1.10%	3.34%
n20k10t5_3		21	**26.09**	26.09	25.17	14400	26.40	26.50	1135.87	1025.78 / 1299.49	1.17%	4.89%
n20k13t5_4			**42.62**	42.62	41.69	14400	42.33	42.85	1251.74	1054.37 / 1514.10	-0.69%	1.54%
n20k12t5_5			**34.07**	34.07	33.80	14400	34.09	34.41	843.64	701.28 / 1152.57	0.06%	0.86%
										Avg.	**0.64%**	**2.81%**
n20k14t5_1			**32.30**	32.30	31.61	14400	32.09	32.37	1093.60	1007.99 / 1209.18	-0.65%	1.52%
n20k10t5_2			**29.37**	29.37	28.78	14400	29.49	29.85	1251.76	1146.36 / 1315.93	0.41%	2.47%
n20k7t5_3	0.50	21	**23.25**	23.25	22.65	14400	23.44	23.53	1859.05	1747.03 / 1971.07	0.81%	3.49%
n20k10t5_4			**24.81**	24.81	24.12	14400	24.80	24.90	1352.20	1123.60 / 1593.77	-0.04%	2.82%
n20k11t5_5			**36.45**	36.45	35.60	14400	36.13	36.59	1044.04	1007.03 / 1102.29	-0.89%	1.49%
										Avg.	**-0.07%**	**2.36%**
n50k10t5_1			**44.65**	44.65	36.04	14400	38.40	38.40	14455.40	14455.40	-16.28%	6.55%
n50k8t5_2			**32.91**	32.91	30.36	14400	32.24	32.24	14446.60	14446.60	-2.08%	6.19%
n50k9t5_3		51	**33.13**	33.13	28.84	14400	31.31	31.31	14452.70	14452.70	-5.81%	8.56%
n50k9t5_4			**29.61**	29.61	27.04	14400	28.79	28.79	14452.70	14452.70	-2.85%	6.47%
n50k11t5_5			**39.29**	39.29	31.04	14400	33.67	33.67	14430.20	14430.20	-16.69%	8.47%
	0.15									**Avg.**	**-8.74%**	**7.25%**
n100k17t5_1			—	—	61.78	14400	67.21	67.21	14549.10	14549.10	—	8.79%
n100k18t5_2			—	—	72.28	14400	77.60	77.60	14545.10	14545.10	—	7.36%
n100k20t5_3		101	—	—	75.39	14400	81.01	81.01	14529.90	14529.90	—	7.45%
n100k21t5_4			—	—	68.57	14400	74.57	74.57	14539.70	14539.70	—	8.75%
n100k18t5_5			—	—	84.44	14400	91.21	91.21	14503.10	14503.10	—	8.02%
										Avg.	—	**8.07%**
										Total Average Gap	**-0.30%**	**3.83%**

[1] Not considered for average calculations.

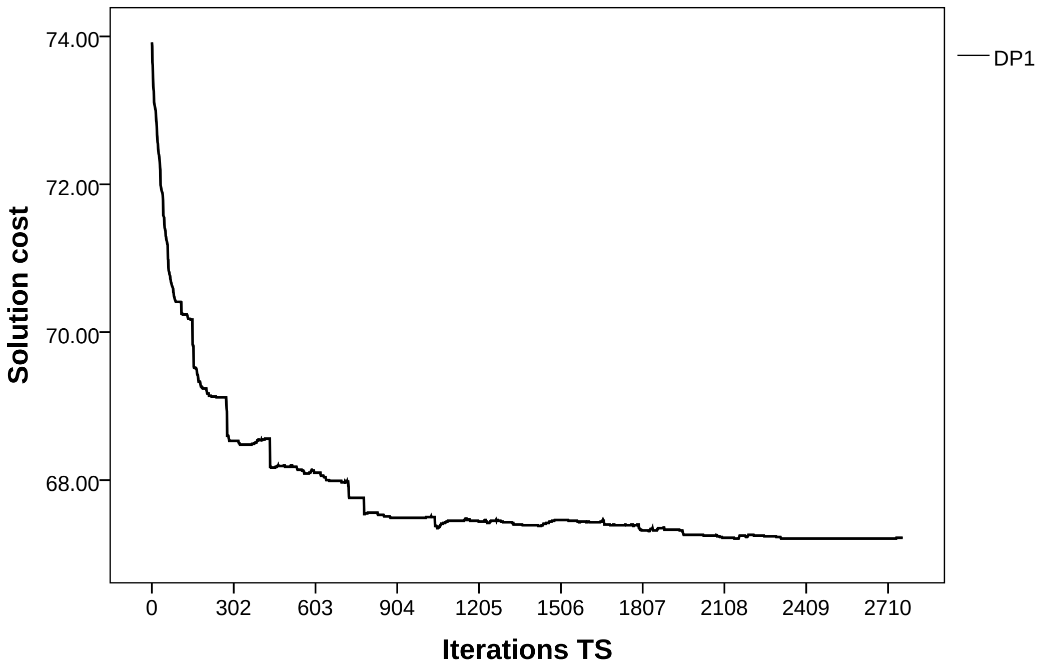
Table 4: Performance of the two-phase algorithm.

values indicate that the two-phase algorithm produces high quality solutions. In addition, the lower bounds produced by the FPVRP formulation of Archetti et al. (2017) are, in general, pretty tight. No LB gap is reported for instance p32, since the FPVRP formulation was not able to obtain a lower bound for that instance. In general, the computing times required by the heuristic are moderate (requiring a total average time of 6480.5 sec for S1 and 659.15 sec for S2) taking into account the difficulty and dimensions of the considered instances. Furthermore, it was able to find good quality feasible solutions for all the considered benchmark instances, whereas the FPVRP formulation was not. We have observed that increasing the `IterNImp` value may produce improvements in the solution quality of the small size instances.

We conclude this section by illustrating the evolution of the optimization process of two of the large instances, one with 50 customers and one with 100 customers. Figure 3 shows the decrease of the value of the objective function of the solution found by the TS for the two considered instances. Figure 3a refers to the instance with 50 customers while Figure 3b is related to the instance with 100 customers. On the vertical axis we report the value of the objective function found by the TS while on the horizontal axis we have the number of iterations performed by the TS. In particular, for the instance with 50 customers, we have 4 DP generations (DP1, DP2, DP3 and DP4 in Figure 3a) after the maximum computing time is reached. For the instance with 100 customers, the maximum computing time is reached during the first run of the TS. We observe that most of the improvements occur before iteration 1000, which is reached after 4290 seconds for the 100 customer instance. For the 50 customer instance, 1000 iterations are reached after 774.63 seconds for DP1, 749.08 seconds for DP2, 772.87 seconds for DP3, and 622.52 for DP4. Note that, for the 50 customer instance, the incumbent solution, from the first iteration, is better than the solution produced by the FPVRP formulation after 4 hours of computation. For the 100 instance no comparison can be done as the formulation does not produce any feasible solution within 4 hours of computing time.



(a) 50 customers (n50k10t5_1)



(b) 100 customers (n100k17t5_1)

Figure 3: Value of the incumbent solution for a 50/100 customers instance.

5 Conclusions and Future Research

In this paper we have presented a two-phase algorithm for the Flexible Periodic Vehicle Routing Problem (FPVRP), which deals with periodic demands over a planning horizon. In the FPVRP, each customer has a total demand that has to be satisfied by the end of the planning horizon. The quantity delivered at each visit should not exceed the customer storage capacity, which is typically lower than the total demand. Thus, multiple visits to each customer must be performed over the planning horizon. The FPVRP can be seen as an extension of the Periodic Vehicle Routing Problem and it is also related to the Inventory Routing Problem. The FPVRP was introduced in Archetti et al. (2017) where MILP formulations were proposed, which allowed to solve instances with up to 20 nodes.

The algorithm proposed in this paper consists of two phases, which alternate iteratively. The first phase builds an initial solution by first obtaining a distribution plan through the solution of a MILP, in which the routing costs are approximated, and then generating a set of feasible routes, consistent with the distribution plan, is built through the solution of a series of VRPs for each time period. The second phase is a Tabu Search heuristic, which explores several neighborhoods, and aims at improving the solution produced by the first phase.

Extensive computational experiments have been run with benchmark instances with up to 100 customers. Best-known solutions are obtained with a small percentage deviation to the lower bounds produced by the formulation. The results obtained show that the heuristic can solve the considered instances with an average gap from best-known solutions of -0.30%, where a negative value indicates that the proposed algorithm outperforms the exact solution. Furthermore, for instances with 50 customers, the proposed approach was able to obtain better solutions than the FPVRP formulation in much less computing time while for instances with 100 customers, for which the FPVRP formulation is not able to produce any feasible solution, our algorithm produces solutions with an average gap of 8.07% from the lower bound. Therefore, the obtained results assess the effectiveness of the heuristic, particularly for the largest instances, which cannot be handled with the existing formulations.

Acknowledgements

The research of the last two authors has been partially supported by the Spanish Ministry of Economy and Competitiveness end EDRF funds through grants MTM2015-63779-R and MDM-2014-044 (MINECO/FEDER). The research of the third author has been partially supported by the Mexican National Council for Science and Technology (CONACyT) and *La Caixa* Foundation. This support is gratefully acknowledged. Furthermore, the authors are grateful to the reviewer for the valuable comments, which have led to a considerable improvement of the manuscript.

References

Archetti, C., Bertazzi, L., Hertz, A., and Speranza, M. G. (2012). A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116.

Archetti, C., Fernández, E., and Huerta-Muñoz, D. L. (2017). The flexible periodic vehicle routing problem. *Computers & Operations Research*, 85:58–70.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valleta, A. (2011). An exact algorithm for the period routing problem. *Operations Research*, 59(1):228–241.

Bertazzi, L. and Speranza, M. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1:307–326.

Bertazzi, L. and Speranza, M. (2013). Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, 2:255–275.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, 63(1):2–15.

Chao, I.-M., Golden, B. L., and Wasil, E. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, 26(1):25–44.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Cordeau, J., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

Francis, M. P., Smilowitz, K., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In Golden, B., raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer, New York.

Glover, F. W. and Laguna, M. (1997). *Tabu Search.* Number 1. Springer US, New York.

Groër, C., Golden, B., and Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2:79–101.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516.

Ralph, T. K., Kopman, L., Pulleyblank, W. R., and Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2–3):343–359.