Computers & Operations Research 53 (2015) 9–23

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm

Alireza Rahimi-Vahed [a,*], Teodor Gabriel Crainic [b], Michel Gendreau [c], Walter Rei [b]

[a] *Département Informatique et recherche opérationnelle and Interuniversity Research Centre on Enterprise Networks, Logistics, and Transportation (CIRRELT), Université de Montréal, C.P. 8888, succ. Centre-ville, Montréal, QC, Canada H3C 3P8*
[b] *Département Management et technologie, ESG and Interuniversity Research Centre on Enterprise Networks, Logistics, and Transportation (CIRRELT), Université du Québec à Montréal, C.P. 8888, succ. Centre-ville, Montréal, QC, Canada H3C 3P8*
[c] *Département de mathématiques et génie industriel and Interuniversity Research Centre on Enterprise Networks, Logistics, and Transportation (CIRRELT), École Polytechnique, C.P. 6079, Succ. Centre-ville, Montréal, Canada H3C 3A7*

A R T I C L E I N F O

Available online 15 July 2014

Keywords:
Multi-depot periodic vehicle routing problem
Fleet-sizing
Modular heuristic algorithm

A B S T R A C T

In this paper, we address the problem of determining the optimal fleet size for three vehicle routing problems, i.e., multi-depot VRP, periodic VRP and multi-depot periodic VRP. In each of these problems, we consider three kinds of constraints that are often found in reality, i.e., vehicle capacity, route duration and budget constraints. To tackle the problems, we propose a new Modular Heuristic Algorithm (MHA) whose exploration and exploitation strategies enable the algorithm to produce promising results. Extensive computational experiments show that MHA performs impressively well, in terms of solution quality and computational time, for the three problem classes.

© 2014 Published by Elsevier Ltd.

1. Introduction

In the classical vehicle routing problem (VRP), a homogeneous fleet of vehicles services a set of customers from a single distribution depot or terminal. Each vehicle has a fixed capacity that cannot be exceeded and each customer has a known demand that must be fully satisfied. Each customer must be serviced by exactly one visit of a single vehicle and each vehicle must depart from the depot and return to the depot [1].

Several variations and specializations of the VRP, ranging from the Capacitated VRP (CVRP) to more complex problems including various realistic attributes and constraints, have been extensively studied in the past five decades. However, the literature underlines that there still exist many VRP variants that have not received adequate attention despite their important relevance to many applications.

Among such VRP variants, the Multi-Depot Periodic VRP (MDPVRP) is the one which has not been studied too much, relative to the other existing variants, despite the fact that it reflects many real-life applications (e.g., food collection and distribution [2], maintenance operation [3], and raw material supply [4]). In this paper, we contribute toward addressing this challenge by studying a variant of the MDPVRP, which we refer to as the fleet-size Multi-Depot Periodic VRP (fs-MDPVRP), and two of its special cases, the fleet-size Periodic VRP (fs-PVRP) and the fleet-size Multi-Depot VRP (fs-MDVRP).

The fs-MDPVRP studied in this paper, contrary to the classical MDPVRP in which the goal is often to minimize the total travel distance (cost), seeks to determine the optimal number of vehicles needed for delivery operations over a given planning horizon. However, the fs-MDPVRP shares the network structure and several characteristics with the classical MDPVRP. More precisely, in this VRP, it is assumed that there exists a finite number of depots where vehicles are located. Each vehicle performs only one route per period and each vehicle route must start and finish at the same depot. Each customer may require to be visited on different periods during the planning horizon and these visits may only occur in one of a given number of allowable visit-period combinations. In this problem, as mentioned, the goal is to determine the optimal fleet size where three practical constraints, i.e., vehicle capacity, maximum route duration and budget constraints, should be satisfied. A fs-MDPVRP incorporating such characteristics reduces to a fs-PVRP or a fs-MDVRP if the number of depots or periods is set to 1.

To the best of our knowledge, there is no significant contribution in the literature to address the above VRPs. In this paper, to tackle each of the considered problems, we propose a heuristic solution method that builds a solution at each iteration using a procedure that has three phases, each focusing on one of the decisions to be made (the selection of periods when each customer will be served, the assignment of customers to depots, and the design of routes). The proposed heuristic algorithm incorporates different exploration and exploitation strategies to produce good results, in terms of solution

* Corresponding author.
E-mail address: alireza.rahimi.vahed@umontreal.ca (A. Rahimi-Vahed).

http://dx.doi.org/10.1016/j.cor.2014.07.004
0305-0548/© 2014 Published by Elsevier Ltd.

quality and computational efficiency. The proposed heuristic solution method is numerically shown to be efficient to address a wide variety of test problems for the three problem classes.

The remainder of this paper is organized as follows: Section 2 gives the problem statement. In Section 3, the literature survey relevant to the topic of this study is presented. Different aspects of the proposed heuristic algorithm are described in Section 4. The experimental results are given in Section 5. Finally, Section 6 provides conclusions and the evaluation of the work.

2. Problem statement and modeling

In this section, we formally state each of the problem classes, introducing the notations used throughout this paper.

fs-MDVRP: Consider an undirected graph $G(V,E)$. The node set V is the union of two subsets $V = V_C \cup V_D$, where $V_C = \{C_1, \ldots, C_N\}$ represents the customers and $V_D = \{D_1, \ldots, D_M\}$ includes the depots. With each node $i \in V_C$ is associated a deterministic demand q_i. The edge set E contains an edge for each pair of customers and for each depot-customer combination. There are no edges between depots. With each edge $(v_i, v_j) \in E$ is associated a travel distance d_{ij} [5].

fs-PVRP: In the fs-PVRP, the undirected graph $G(V,E)$ is modified by fixing the value of M to one and by introducing a planning horizon of T periods. In such a graph, each customer i is characterized by a service frequency f_i, stating how often within these T periods the customer must be visited and a list L_i of possible visit-period combinations [6].

fs-MDPVRP: Finally, the fs-MDPVRP combines the two above problem settings, asking for the selection of a depot and a visit pattern for each customer, with services in different periods to the same customer being required to originate at the same depot [6].

In each problem class, the goal is to minimize the maximum number of vehicles used over the planning horizon. By considering such an objective function, we actually address the strategic version of the problem in which we assume that vehicles are either bought or rented over a long period of time by the company facing the problem.

To model the above problems, we consider the three following restrictions which reflect important requirements that are often found in real-life applications:

1. *The vehicle capacity constraint*: This constraint states that the total demand of the customers on any route should not exceed the vehicle capacity Q.
2. *The route duration constraint*: This constraint ensures that the total duration of a route does not exceed a preset value D.
3. *The budget constraint*: In many logistical systems, one is usually faced with budgetary constraints that come from the fact that a limited investment budget is available for a certain area or a certain period of time. Budget considerations are almost always ignored when dealing with VRPs. In this paper, we consider such a budgetary restriction which we refer to as the Travel-Distance Budget (TDB) constraint. The TDB constraint reflects many real-life cases (for example, garbage collection and milk distribution systems) in which, due to limited financial resources to completely cover system's operating costs (e.g., high fuel price and vehicle depreciation costs), vehicles cannot be practically allowed to travel more than a prespecified distance. In this study, the TDB constraint is defined using two different models, each realizing an important managerial challenge in real-life distribution and logistical systems. In the first model (R_1), we set a bound on the total distance that vehicles are permitted to travel over the planning horizon. On the other hand, the second model (R_2) aims to reflect the situations in which, due to geographical and operational constraints, the total distance traveled by vehicles assigned to a depot cannot exceed an imposed limit in each period.

Depending on how we model the TDB constraint, each of the above problem classes can be expressed by one of the following mathematical programming models:

$$(R_1) \min K \tag{1}$$

subject to

$$F(x,K) \le b \tag{2}$$

$$\tau \le \epsilon \tag{3}$$

where K is the maximum required number of vehicles (fleet size) needed over the planning horizon. More precisely, K is defined as the sum of the vehicles needed at each depot. Constraint (2) corresponds to the vehicle-capacity and route-duration restrictions described above. Constraint (3) imposes that the total traveled distance (τ) is limited by a positive value ϵ:

$$(R_2) \min K \tag{4}$$

subject to

$$F(x,K) \le b \tag{5}$$

$$\tau_{tj} \le \epsilon_{tj} \quad \forall t \in T, \quad \forall j \in D; \tag{6}$$

where τ_{tj} is the total distance traveled by the vehicles assigned to depot j in period t and ϵ_{tj} is a positive upper bound which is set on τ_{tj}.

Ref. [5] showed that the formulation of a generalized PVRP includes the MDVRP as a special case by associating a different period to each depot, such that the ith customer has a frequency $f_i = 1$ and can be visited in any period. Ref. [6] extended this result by proving that a MDPVRP with T periods and D depots can be transformed into a generalized PVRP by associating a period to each (period, depot) pair, such that the ith customer, having a list L_i of patterns, is visited f_i times over the planning horizon using one of the $D \times L$ patterns. We rely on these two transformations in the development of the proposed modular heuristic algorithm.

3. Literature review

In this section, we focus on reviewing papers formerly published in the literature to address different settings of the MDPVRP and its two special cases, i.e., the PVRP and the MDVRP. The goal of this review is first to present the most recently proposed heuristic and meta-heuristic algorithms for these VRPs, and to highlight that there is no significant contribution dealing with the problem settings considered in this paper.

To the best of our knowledge, all the MDVRPs and PVRPs studied in the literature consider the total travel distance (cost) as the main goal, regardless of constraint types that they deal with. The majority of solution methods, put forward to address these problems, are divided into (1) classical heuristics which range from simple construction and improvement procedures [7–11] to more structured algorithms as iterative heuristics [12–14] and multi-phase solution methods [15–17], and (2) meta-heuristics which clearly outperform the classical heuristics by benefiting of better exploitation and exploration strategies. In a general point of view, these methods are grouped into tabu search algorithms [18,5], variable neighbourhood search [19,20], large-scale neighbourhood search algorithm [21], and evolutionary meta-heuristics as genetic algorithm [22,6], ant colony optimization [23], and scatter search [4].

Similar to the above problem settings, all the MDPVRPs existing in the literature address the problem of minimizing the total travel distance (cost). Solution methods, targeting these MDPVRPs, are divided into two main groups: (1) classical heuristics, which often address the problem in a sequential manner, and (2) meta-heuristics and parallel algorithms, which tackle the problem by simultaneously optimizing all the involved attributes.

We are aware of three heuristics in the first group. Ref. [3] formulated the problem of providing maintenance services to a set of customers as the MDPVRP with Time Windows (MDPVRPTW). The authors proposed a multi-phase optimization problem and solved it using a four-phase algorithm. In the developed algorithm, all customers are first assigned to a particular depot. Then, customer visits are successively inserted among available periods to obtain feasible visit combinations. In the third phase, each of the depot-period VRP sub-problems is separately solved using a tabu search algorithm. Finally, in the last phase, solutions obtained during the optimization process are improved by modifying the period or depot assignments through a 3-opt procedure. Ref. [24] designed a two-phase heuristic method to address the same problem. In the proposed method, all feasible schedules are first generated from each depot for each period and, then, the set of routes are determined through using the shortest path problem. Ref. [2] proposed a tabu search heuristic to tackle the problem considered by [3]. In this algorithm, all the initial assignments are built by cheapest insertion, where each customer is assigned to its nearest depot and is given its most preferred visit pattern. In the improvement phase, a neighbourhood search is defined by depot and visit pattern interchanges.

We are also aware of three contributions belonging to the second group. The first contribution was the evolutionary metaheuristic proposed by [6]. The authors developed a hybrid Genetic Algorithm (GA) to tackle the MDPVRP and two of its special cases, i.e., the Multi-Depot VRP (MDVRP) and the Periodic VRP (PVRP). The most interesting feature of the proposed GA is a new population-diversity management mechanism which allows a broader access to reproduction, while preserving the memory of what characterizes good solutions represented by the elite individuals of the population. The second contribution was the cooperative parallel algorithm designed by [25]. The authors proposed a structured cooperative parallel search method, called the Integrative Co-operative Search (ICS), to solve highly complex combinatorial optimization problems. The proposed ICS framework involves problem decomposition by decision sets, integration of elite partial solutions yielded by the sub-problems, and adaptive guiding mechanism. The authors used the MDPVRPTW to present the applicability of the developed methodology. Ref. [26] designed a path relinking algorithm to tackle the MDPVRP. The proposed algorithm was designed based on different exploration and exploitation strategies which enable the algorithm to address the problem in two different ways: (1) as a pure stand alone algorithm and (2) as an integrator in the ICS solution framework.

This literature review supports the general statements made in Section 1 that different variants of the MDPVRP, including the MDVRP and the PVRP, are among the VRPs which did not receive an adequate degree of attention and the solution algorithms proposed to address them are scarce. Moreover, there is no significant contribution in the literature dealing with the minimization of the fleet size for the problem settings considered in this paper. To contribute toward addressing these three challenges, we develop a Modular Heuristic Algorithm (MHA) to efficiently address the fs-PVRP, the fs-MDVRP and the fs-MDPVRP. The proposed MHA is described in the next section.

4. The proposed modular heuristic algorithm (MHA)

In this section, we propose a new modular heuristic algorithm which addresses each of the considered problems in three sequential phases, each targeting one specific dimension of the problem. The general concepts and structure of the heuristic algorithm are first described in Section 4.1. Then, the main components of the MHA are explained in detail in Sections 4.2–4.6.

4.1. The general structure of MHA

The Modular Heuristic Algorithm (MHA) that we propose is based on the sequential optimization paradigm, but it includes a number of advanced exploration and exploitation features which contribute to its high performance level, proved by our experimental results shown in Section 5, in terms of solution quality and computational efficiency.

The general scheme of the proposed heuristic algorithm is displayed in Algorithm 1. MHA is an iterative heuristic where, at each iteration, three different phases that sequentially address the decisions to be made, should be executed. These decisions are the visit pattern assignment, the depot assignment and the detailed route design. These three phases can be performed in different orders depending on the problem in question. Our pre-testing experiments show that the order generating the best results, in terms of solution quality and computational efficiency, is (see Section 5.3):

1. The visit pattern assignment: In this phase, each of the customers is assigned to one of the possible visit patterns.
2. The depot assignment: In this phase, customers of each period are assigned to depots.
3. The routing problem: Finally, in this phase, the routes are established for each period and depot.

In the following sections, we thoroughly describe how the interaction between these three phases may contribute to reducing the fleet size and possibly, the total travel distance. Note that the proposed MHA terminates its searching process if one of the following termination criteria is met:

- MHA is stopped if no improving solution is found for Ψ successive iterations. Ψ is a positive value which is determined at the beginning of the algorithm. Or,
- MHA is terminated if it passes a maximum allowable running time.

Algorithm 1. Modular heuristic algorithm.

```
-Initialize the search parameters.
-Set the initial reference set as an empty list.
while the termination criterion is not met do
    -Assign a possible visit pattern to each customer.
    -Assign each customer to a depot in each period of the
     selected visit pattern.
    -Design routes visiting customers assigned to the same depot
     using the three-phase heuristic.
    -Update the reference list.
end while
```

In the following sections, the main components of the proposed MHA are described in detail.

4.2. Solution representation

The first step in designing an algorithm for a particular problem is to devise a suitable solution representation scheme. In the proposed heuristic algorithm, the path representation proposed by [26] is used. The idea of this path representation is that the customers are listed in the order in which they are visited. In this kind of representation, a single row array of the size equal to $N+1$ is generated for each depot in each period, where N is the number of customers to be visited. The first position of the array (index 0) is related to the corresponding depot, while each of the other positions (index i; $1 \le i \le N$) represents a customer. Each position of the array can take any integer value from the interval $[-N, N]$. This integer value represents the customer who should be immediately visited after the customer or depot related to that position. In this path representation, negative values indicate the beginning of a new route, 0 refers to the end of the routes and a vacant position reveals that the customer corresponding to that position is not served by the depot to which the array belongs. Using this representation, changes to the solution can be performed very quickly. For example, the insertion of a new customer k between two adjacent customers a and b is done simply by changing the "next-values" of k to b and a to k. Similarly, one can delete a customer or reverse part of a route very quickly. For a detailed description of the above solution representation, readers should refer to [26].

4.3. Elitism strategy

One of the most important characteristics of MHA is to use an elitism strategy which enables the algorithm to retain good and diverse solutions obtained in the course of the optimization. Towards this end, MHA keeps high-quality and diverse solutions in a list called the reference set. The reference set has the size equal to a predetermined positive value γ and consists of two different subsets. The first subset, B_1, preserves $\lceil 3 \times \gamma/4 \rceil$ high quality solutions, while the second subset, B_2, is made up of $\lfloor \gamma/4 \rfloor$ diverse solutions. It is worth mentioning that the size of each subset has been determined after running an extensive pre-testing experiments. The reference set is initially empty and is iteratively updated. Suppose a new solution, S_{new}, is obtained by the algorithm. The reference set is updated using the following two steps:

1. S_{new} is first investigated in terms of solution quality. In this case, S_{new} is directly added to B_1 if the number of solutions preserved in B_1 is less than $\lceil 3 \times \gamma/4 \rceil$; otherwise, if S_{new} is better than the worst existing solution in B_1, the latter is replaced by the former.
2. If none of the conditions mentioned in Step 1 is met, S_{new} is assessed in terms of solution diversification. In this case, S_{new} is directly added to B_2 if the number of solutions existing in B_2 is less than $\lfloor \gamma/4 \rfloor$; otherwise, the following replacement strategy is implemented. We first define the contribution to diversity of solution S_{new} to the first subset of the reference set, $D(S_{new}, B_1)$, as the similarity between itself and its nearest neighbour in B_1, that is

$$D(S_{new}, B_1) = \min_{X \in B_1, X \ne S} \Delta(S_{new}, X)$$

where $\Delta(S_{new}, X)$ is the Hamming distance. Moreover, let us define $OF_{S_{new}}$ as the objective function value of solution S_{new}. The replacement strategy is implemented in three phases as follows: firstly, the replacement strategy considers all the solutions of B_2 with poorer objective function values than S_{new} and finds the one, S_{max}, which maximizes the ratio of (*objective function value*)/(*contribution to diversity*) (Step 1). Then, the new generated solution, S_{new}, replaces S_{max} if the following inequality holds (Step 2):

$$\frac{OF_{S_{new}}}{D(S_{new}, B_1 - S_{max})} < \frac{OF_{S_{max}}}{D(S_{max}, B_1)} \tag{7}$$

In this way, we introduce into B_2 a solution with better objective function value and possibly higher contribution to diversity. If inequality (7) does not hold, the worst solution of the set determined in the first step is replaced by S_{new} (Step 3).

It is worth mentioning that two special variants of MHA can be obtained by considering only B_1 and B_2, respectively. In the former case, MHA keeps only high-quality solutions generated in the course of optimization, while in the latter case, only high-diverse solutions are gathered in the reference set. Our extensive pre-testing experiments show that simultaneously considering both B_1 and B_2 in the reference set results in a more efficient algorithm, in terms of solution quality and computational efficiency.

4.4. Visit pattern assignment

The modular heuristic, as depicted in Algorithm 1, first assigns, at each iteration, a possible visit pattern to each customer. There exist a very scarce number of contributions in the literature that use systematic and non-random methods to assign customers to visit patterns. For example, [27] extended the generalized assignment problem of [28] to assign a customer to an allowable visit pattern. Ref. [29] developed a median problem to establish an initial assignment of customers to visit patterns that meets customer service requirements. Ref. [16] designed a generalized network approximation method to assign visit patterns to customers. The proposed method was represented by a tripartite transshipment graph with source nodes (customers), transshipment nodes (allowable visit patterns), and sink nodes (periods of the planning horizon).

In this study, we also propose a non-blind and non-random strategy to systematically assign customers to their possible visit patterns. In the proposed algorithm, we first locate a single point, called reference point, for each period of the planning horizon and, then, using a new Integer Programming Model (IPM), customers are assigned to visit patterns so as to minimize the sum of customer-to-reference point distances. The main idea of using such a customer-visit pattern assignment procedure is to create well-packed clusters of customers over different periods of the planning horizon which may result in reducing the fleet size needed for delivery operations. It is worth mentioning that the reference point considered in this study is different from a typical seed point that is often used, in the VRP literature, to build the routes of a depot (for more information about the concept of a seed point, readers are referred to [30]). In this case, the reference point is an indicator which determines, using a predefined manner, what customers should be visited together in a given period of the planning horizon. In the following, we describe how efficient customers-visit pattern assignments can be obtained by purposefully positioning such reference points in the different considered periods.

The proposed IPM, unlike the similar models existing in the literature, is governed by parameters whose values are dynamically adjusted in the course of the search process. Our pre-testing experiments show that this feature enables the IPM to assign better visit patterns to customers as MHA evolves.

The integer programming model is formulated as follows:

$$\min \sum_{i=1}^{N} \sum_{k \in L_i} \left[\sum_{t=1}^{T} a_{kt} d_{ia_t^\nu} \right] u_{ik} \tag{8}$$

subject to

$$\sum_{k \in L_i} u_{ik} = 1 \quad \forall i \in V; \tag{9}$$

$$LB_t^\nu \leq \sum_{i=1}^{N} \sum_{k \in L_i} a_{kt} q_i u_{ik} \leq UB_t^\nu \quad \forall t \in T; \tag{10}$$

$$u_{ik} \in \{0,1\} \quad \forall i \in V, \quad \forall k \in L_i; \tag{11}$$

In the above model, $d_{ia_t^\nu}$ shows the distance between customer i and reference point a which is generated, in iteration ν of the algorithm, for period t. Moreover, the parameter a_{kt} equals 1 if period t is in pattern k, and 0 otherwise. The decision variable of this model is u_{ik}, which is equal to 1 if customer i is assigned to pattern k, and 0 otherwise. Finally, LB_t^ν and UB_t^ν are respectively lower and upper limits that bound the total number of demands which should be satisfied in period t and iteration ν. As it can be seen in this mathematical formulation, the aim is to assign possible visit patterns to customers so that the total squared Euclidean distance between the customers and reference points is minimized. Note that other distance functions, i.e., shortest path between points in a road network, can also be used. In other words, the IP model simply requires parameters $d_{ia_t^\nu}$ to be fixed according to the specific function that is used. Constraints (9) impose that each customer is assigned to exactly one feasible pattern. Constraints (10) show that the demands serviced in a given period must be within an imposed interval.

One of the most crucial parameters affecting the effectiveness of the integer programming model is the reference points' locations. In this paper, the reference points' locations are generated using a memory-based algorithm as follows: we first enumerate all the customers that can be serviced in period t (t=1,2…T). That is, $I_t = \{i \in V_c | \exists k \in L_i : a_{kt} = 1\}$. Then, the coordinates of the reference point are calculated using the two following formulas:

$$x_t^\nu = \frac{\sum_{i \in I_t} w_{it}^\nu x_i}{\sum_{i \in I_t} w_{it}^\nu} \quad \forall \nu = 1,2\ldots, \quad \forall t \in T; \tag{12}$$

$$y_t^\nu = \frac{\sum_{i \in I_t} w_{it}^\nu y_i}{\sum_{i \in I_t} w_{it}^\nu} \quad \forall \nu = 1,2\ldots, \quad \forall t \in T; \tag{13}$$

In the above equations, (x_i, y_i) represents the position of customer i. Moreover, w_{it}^ν is defined as a self-adjusting positive weight which reflects the desirability of visiting customer i in period t at iteration ν of the algorithm. The values of these weights are dynamically adjusted, at each iteration, based on the information gathered from the reference set. The adjusting procedure used in this study is presented by the following formula:

$$w_{it}^\nu = \begin{cases} (1+\varphi) \times w_{it}^{\nu-1} & \text{If customer } i \text{ is visited in period } t \text{ and iteration } \nu-1 \\ & \text{in more than } \theta\% \text{ of the solutions existing in the} \\ & \text{reference set} \\ w_{it}^{\nu-1} & \text{Otherwise} \end{cases}$$

where φ and θ are two predefined positive parameters. It should be noted that this adjusting procedure starts after the modular heuristic passes a preliminary phase called Warming-up Stage (WS). In WS, the algorithm is repeated in λ successive iterations, $\lambda \geq \gamma$, so that, at each iteration, the weights involved in Eqs. (12) and (13) are randomly generated in the interval (0,1].

The other important parameters having key roles in the integer programming model are the bounds considered in constraints (10). In this paper, LB_t^ν and UB_t^ν define parameters which are iteratively adjusted based on the information obtained from elite solutions kept in the reference set. Towards this end, LB_t^ν and UB_t^ν are respectively defined, in iteration ν, as the minimum and maximum required capacity for period t observed in elite solutions existing in the reference set. It should be noted that, in the Warming-up Stage described above, constraints (10) are relaxed by respectively setting LB_t^ν and UB_t^ν to 0 and ∞.

4.5. *Depot assignment*

The proposed heuristic continues by assigning the customers of each period to depots. These assignment decisions are made by solving a clustering problem for each period. In this case, each cluster is defined as a subset of nodes that includes a depot and the customers assigned to it.

To construct compact clusters, the modular heuristic algorithm assigns each customer to its nearest depot according to the weighted squared Euclidean distance with weights b_{ijt}^ν. In this study, b_{ijt}^ν, showing the desirability of assigning customer i, in iteration ν, to depot j and period t, is defined as a self-adjusting parameter whose value is iteratively updated using the information obtained from the reference set. The main goal involved in using this customer-depot assignment procedure is exactly the same as explained in the visit pattern assignment phase. The updating procedure is

$$b_{ijt}^\nu = \begin{cases} (1+\mu) \times b_{ijt}^{\nu-1} & \text{If customer } i, \text{ at iteration } \nu-1, \text{ is assigned to depot } j \\ & \text{and period } t \text{ in more than } \theta\% \text{ of the solutions kept} \\ & \text{in the reference set} \\ b_{ijt}^{\nu-1}/(1+\mu) & \text{Otherwise} \end{cases}$$

where μ is a positive parameter. This updating procedure enables the algorithm to assign customers to better depots as the heuristic gets closer to the termination criteria. It should be noted again that, in the Warming-up Stage, the penalties involved in the objective function are randomly generated in the interval (0,1].

4.6. *Route design*

In this phase, customers assigned to the same depot, in each period, are sequenced into different routes by a three-step heuristic algorithm:

1. *Construction step*: In the first step, customers of each depot are assigned to a giant tour using the GENIUS heuristic to address the corresponding traveling salesman problem. GENIUS, proposed by [31], consists of two phases. In the first phase, called GENI, a Hamiltonian cycle is progressively generated by inserting vertices (i.e., customers) one at a time. More precisely, GENI starts with a partial solution consisting of three arbitrarily chosen vertices and, at each iteration, it includes any given vertex between two of its p closest neighbors on the partial cycle. While making an insertion, GENI performs a local reoptimization of the partial cycle. Once all vertices have been inserted, the second phase, named US, is executed as a post-optimization heuristic which successively removes each vertex from the cycle, and reinserts it using GENI.
2. *Splitting step*: In the second step, using the optimal splitting procedure proposed by [32], each constructed giant tour is split into shorter routes, each satisfying the vehicle capacity constraint. In other words, by relaxing the route duration restriction and budget constraint, each giant tour and, consequently, the whole solution are split to least possible number of shorter routes.
3. *Improving step*: Finally, in the third step, a heuristic method, represented by Algorithm 2, is implemented on each constructed route in order to reduce the route's length and, accordingly, to improve the total traveled distance. One of the main features of the proposed heuristic method is that the search is allowed to enter infeasible regions. In fact, this feature enables the heuristic method to explore different regions (feasible and infeasible) which may result in generating high-quality and diverse feasible solutions. Let us assume that X denotes the new solution generated by the search mechanism.

Moreover, let $\tau(X)$ denote the total traveled distance of solution X, and let $A(X)$ and $B(X)$ denote the total violation of the route-length and budget constraints, respectively. Solution X is evaluated by a penalty function, $z(X) = \tau(X) + \alpha A(X) + \beta B(X)$, where α and β are self-adjusting positive parameters. By dynamically adjusting the values of these two parameters, this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances. In the following, we describe how α and β are self-adjusted in the course of the improving step. The proposed heuristic method consists of the two following exchange procedures which are implemented in a sequential manner:

- *Swap procedure*: The first exchange procedure, called swap, is performed by choosing two distinct customers i and j, where $i = 1,2 \ldots n_k - 1$ and $j = i+1, i+2 \ldots n_k$ (n_k is the number of customers that are visited in the kth route), and then exchanging their positions within the route. If the swapping procedure results in a better solution according to the penalty function described above, the positions are exchanged, otherwise the solution remains unchanged. In the case where a new solution is generated, parameters α and β are updated as follows: if there is no violation of the route-length constraints, the value of α is divided by $1+\Lambda$, otherwise it is multiplied by $1+\Lambda$, where Λ is a positive parameter. A similar model applies also to β with respect to the budget constraint. The swap procedure stops when no more exchanges that result in an improving solution are possible.
- *Insert procedure*: The second exchange procedure, called insert, is based on removing a customer from one route and inserting it into another route. Towards this end, we first randomly select a customer from the longest route whose length is defined as t_{max}. Then, two insertion methods compete each other to determine the best position for re-inserting the removed customer. These two insertion methods are summarized as follows:
  (a) *Intra-depot method*: In this method, the removed customer is re-inserted into either one of the existing routes or a new route of the same depot from which the customer has been removed.
  (b) *Inter-depot method*: In this method, the depot to which the customer is currently assigned is randomly changed to another one and the customer is re-inserted into either one of the existing routes or a new route of the new depot.

In both the above methods, an insertion is called feasible if (i) it does not violate the vehicle capacity constraint, and (ii) it does not produce a route longer than t_{max}. Moreover, the position to which the customer is re-inserted is the one that satisfies the above two conditions and results in the best improvement in the penalty function. It is worth mentioning that, in the case where there exists at least one position improving the current penalty function value, parameters α and β are adjusted using the same model as described in the swap procedure. This insert procedure is stopped if no improving feasible insertion is found for σ successive number of iterations, where σ is a predetermined positive value.

Algorithm 2. Improving step.

```
X ← current solution.
Swap procedure:
for $k = 1 \ldots K$ do
  for $i = 1 \ldots n_k - 1$ do
    for $j = i+1 \ldots n_k$ do
      Select customers $i$ and $j$.
      Generate a new solution, $X'$, by swapping the selected customers.
      if $z(X') \le z(X)$ then
        $X \leftarrow X'$ and $z(X) \leftarrow z(X')$.
        Update $\alpha$ and $\beta$.
      end if
    end for
  end for
end for
Insert procedure:
$i \leftarrow 1$.
while $i \le \sigma$ do
  Randomly remove a customer from the longest route of $X$.
  Generate two new solutions by respectively applying the inter-depot and intra-depots method on $X$.
  if Both solutions are infeasible according to conditions (I) and (II) or non-improving then
    $i \leftarrow i+1$.
  else
    Replace $X$ with the best feasible solution generated.
    Update $\alpha$ and $\beta$.
    $i \leftarrow 1$.
  end if
end while
```

5. Experimental results

In this section, the performance of the proposed MHA is investigated based on three different sets of test problems. The first two sets, each including 15 problem instances, have been developed by [5] for the PVRP and the MDVRP. The last set consists of 15 MDPVRP instances. This set is partitioned into two different parts. In the first part, which includes 10 different test problems (pr01–pr10) developed by [6], the ratio of the number of depots to the number of periods is set to 1. On the other hand, in the second part, containing five new generated problem instances (pr11–pr15), the above-mentioned ratio is varied to investigate how the performance of MHA is affected in cases where the number of depots and periods are unequal. Note that, in all the considered problem instances, the number of vehicles assigned to a depot, which was originally set as a fixed parameter, is ignored. Detailed information on these sets are provided in Section 5.3.

The efficiency of the developed heuristic is tested in two different settings that are defined according to how the budget constraint is formulated: either using model R_1 or model R_2 defined in Section 2. Recall that, in model R_1, an upper limit is imposed on the total distance traveled over the planning horizon. As for model R_2, an upper bound is enforced on the total distance traveled by the vehicles assigned to each depot in each period.

In both the above cases, values assigned as the upper bound of the budget constraint may have a major impact on the performance of the proposed modular heuristic algorithm. In this paper, the upper bounds of both models (R_1 and R_2) are initially set based on the information extracted from the Best Known Solutions (BKS), which were identified overall runs performed by the Hybrid Genetic Algorithm (HGA) proposed by [6] for the classical version of the problems considered in this paper. More precisely, the values of ϵ, in model R_1, are set to the total traveled distance of the BKS, whereas the value of ϵ_{td}, in model R_2, is fixed to the total distance traveled by the vehicles assigned to depot d in period t of the BKS. Then, we systematically vary the values of the upper bound, set on the budget constraint, to investigate how the performance of MHA is affected by tightening or widening the budget constraint.

The proposed algorithm is run on each problem instance, for both budget constraint models, and its efficiency, in terms of solution quality and computational time, is compared to a modified version of the Generalized Unified Tabu Search (GUTS) implemented in [33]. The original version of GUTS has been shown to be an efficient algorithm to solve the classical version of the problems considered in this study, i.e., the PVRP, the MDVRP, and the MDPVRP. In our experiments, we first did the following modifications to make GUTS applicable to the problem classes of this paper:

- The original objective function, which was the minimization of the total traveled distance, was changed to the minimization of the fleet size.
- Using the principles originally defined in [34], the budget constraints in our models were relaxed and added as penalty terms to the objective.
- The position to which a removed customer is reinserted was determined using the new penalty function. Following this manner, the insertion operator progressively aims to decrease the routes' lengths and possibly, to remove them.

Then, in order to enhance the quality of this algorithm to efficiently deal with the fleet size objective function and, consequently, to have a fair comparison with the proposed modular heuristic, all the involved parameters were once again systematically tuned using the calibration method of [35].

To investigate how successful this vehicle elimination scheme is, a set of experiments was carried out. Towards this end, five instances, each including many customers per route, i.e., pr02, pr04, pr06, pr09, and pr12, were selected. Each experiment was independently run for each pair (problem class, budget constraint formulation) and the results were compared, in terms of the total number of used vehicles, to the case where the budget constraint violations are excluded from the penalty function (shown as GUTS*). It should be noted that, in each of these problem instances, the value of ϵ was set to $0.8\epsilon^*$, where ϵ^* refers to the total traveled distance of the BKS. Our experimental results showed that the average percentage gaps of our modified GUTS with respect to GUTS*, in model R_1, are -2.42%, -3.13%, and -3.94%, for the fs-PVRP, fs-MDVRP, and fs-MDPVRP, respectively. These results reveal the fact that the modified GUTS produces better results compared to GUTS*. The main conclusions derived from model R_2 are similar to those stated above for model R_1. The modified GUTS, once again, outperforms, in average, GUTS* on the considered problem instances. In this case, the average percentage gaps change to -3.09%, -4.44%, and -4.77%, for the fs-PVRP, fs-MDVRP, and fs-MDPVRP, respectively. More detailed results are available upon request.

Different aspects of the experimental results are discussed as follows: in Section 5.1, we first use a well-structured algorithm to calibrate all the parameters involved in the heuristic algorithm. Then, in Section 5.2, a comprehensive sensitivity analysis is done to investigate the relative importance of each algorithmic component's performance to the quality of the overall solution framework. Finally, in Section 5.3, computational results are given in detail.

5.1. Parameter setting

Like most heuristic and meta-heuristic algorithms, the proposed MHA has several parameters that need to be tuned before it can reach good results. The problem then turns into finding the best parameter setting for the heuristic to address the considered problems efficiently and timely. Table 1 provides a summary of all the parameters involved in the algorithm.

There are various different methods in the literature to calibrate parameters used in a heuristic or meta-heuristic algorithm. In this paper, we use the calibration method of [26] to tune the parameters used in the proposed modular heuristic algorithm. Ref. [26] designed a hybrid algorithm based on the statistical Design Of Experiments (DOE) that systematically selects high-quality parameter values. The parameter setting procedure has four steps that are implemented in a sequential manner:

Step 1 : A subset of instances to analyze (analysis set) is chosen from the entire set. The instances are selected so that most of the structural differences found in the problem set are represented in the analysis set. In this study, the analysis set is made up of five instances, i.e., pr02, pr04, pr06, pr09, and pr12.

Step 2 : Computational experience is used to select the starting level of each parameter, the range over which each parameter will be varied, and the amount by which each parameter should be changed. Towards this end, [26] used a robust calibration method called Relevance Estimation and VAlue Calibration (REVAC) [36]. Technically, REVAC is a heuristic generate-and-test method that is iteratively searching for the set of parameter vectors of a given Evolutionary Algorithm (EA) with a best performance. For each iteration, a new parameter vector is generated and its performance is tested. Testing a parameter vector is done by executing the EA with the given parameter values and measuring the EA performance. A detailed explanation of REVAC can be found in [36]. Table 2 summarizes the results obtained using REVAC.

Step 3 : Good parameter settings are selected for each instance in the analysis set using fractional factorial design and response surface optimization. In this paper, the half-factorial design is implemented for each selected analysis set.

Step 4 : Finally, in the fourth step, the parameter values obtained in the third step are averaged to obtain high-quality parameter values. The calibration results for each class, along with the final choice of parameter values, are reported in Table 3.

5.2. Sensitivity analysis of algorithmic components

In this section, a set of experiments was carried out to investigate the relative importance of each algorithmic component's performance to the quality of the overall framework. These

Table 1
Parameters of the heuristic algorithm.

Symbol	Description
γ	Maximum size of the reference set
θ	Threshold defined in Sections 4.4 and 4.5
φ	Factor involved in updating w_{it}
λ	Number of times that WS is repeated
μ	Factor involved in updating b_{ijt}
α, β	Self-adjusting parameters in the penalty function
Λ	Factor involved in updating α and β
σ	Number of times that INSERT is repeated
Ψ	Maximum allowable number of non-improving iterations

Table 2
Primary calibration results.

Symbol	Starting level	Range	Changing step
γ	20	[20,60]	5
θ	0.1	[0.1,1]	0.1
φ	1	[1,5]	1
λ	$\lceil \frac{N*M*T}{50} \rceil$	$\left[\lceil \frac{N*M*T}{50} \rceil, \lceil \frac{N*M*T}{5} \rceil\right]$	$\lceil \frac{N*M*T}{5} \rceil$
μ	1	[1,5]	1
α, β	1,1	[1,5], [1,5]	1,1
Λ	1	[1,4]	1
σ	$\lceil \frac{N}{50} \rceil$	$\left[\lceil \frac{N}{50} \rceil, 2N\right]$	$\lceil \frac{N}{5} \rceil$
Ψ	$\lceil \frac{N*M*T}{50} \rceil$	$\left[\lceil \frac{N*M*T}{50} \rceil, N*M*T\right]$	$\lceil \frac{N*M*T}{5} \rceil$

Table 3
Final calibration results.

Symbol	fs-PVRP	fs-MDVRP	fs-MDPVRP	Final choice
γ	40	50	60	50
θ	0.4	0.5	0.6	0.5
φ	2	2	2	2
λ	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$	$\lceil \frac{N*M*T}{5} \rceil$
μ	1	1	1	1
α, β	1, 2	1, 2	1, 2	1, 2
Λ	1	1	1	1
σ	$2N$	$2N$	$2N$	$2N$
Ψ	$N*M*T$	$N*M*T$	$N*M*T$	$N*M*T$

experiments were performed by either changing the order of components or through removing each component in turn and replacing it by an alternative approach. In the following, we describe the set of experiments considered in this section.

- EX01: In the first experiment, the order of components has changed to (1) the depot assignment, (2) the visit pattern assignment, and (3) the routing problem.
- EX02: In this experiment, the proposed integer programming model of Section 4.4 has been modified by randomly choosing the reference point in each period.
- EX03: In the third experiment, each customer is assigned to its nearest depot. In other words, in this experiment, the possibility of assigning a customer to different depots in different periods has been ignored.
- EX04: In this experiment, the heuristic algorithm proposed in the routing problem has been replaced by the well-known savings algorithm of [37].
- EX05: In the seventh experiment, the heuristic algorithm of Section 4.6 has not been permitted to generate infeasible solutions.
- EX06: In this experiment, the heuristic algorithm of Section 4.6 has been modified by removing the inter-depot method from the insert procedure.
- EX07: In the ninth experiment, the inter-depot method has been modified so that the visit pattern and depot of the removed customer are simultaneously changed to new ones.

Each experiment was independently run for each problem class, considering the same five problem instances used to calibrate the parameter settings. In each of these problem instances, the value of ϵ was set to $0.8\epsilon^*$. The results are reported in Tables 4 and 5 for models R_1 and R_2, respectively. In these tables, each column corresponds to the relative differences, in terms of Objective Function Value (OFV) and computational time (Time), with respect to the basic implementation of the modular heuristic algorithm.

Considering Tables 4 and 5, it is noticeable that all the algorithmic components, as well as the implementation order, considered in this paper play an important role in the good performance of the proposed modular heuristic algorithm, the most crucial being the proposed visit pattern and depot assignment mechanisms.

5.3. Computational results

We tested the proposed modular heuristic algorithm on the problem instances described at the beginning of this section using the two models described in Section 2, models R_1 and R_2. Detailed computational results on each fold are given in the following subsections.

5.3.1. Model R_1

In model R_1, both the modular heuristic and GUTS are run 10 times on each problem instance. Moreover, the maximum running time of both algorithms for the fs-PVRP and fs-MDVRP instances is set to 20 min, while for the fs-MDPVRP instances, due to their greater difficulty, the maximum running time is fixed to 30 min.

Results on fs-PVRP instances are presented in Tables 6 and 7. The first five columns of these tables display instance identifier, number of customers, number of depots, number of periods, and the fleet size observed in BKS. Moreover, in column 6, different values of the upper bound, set on the budget constraint, are shown. The results of the proposed heuristic method are shown in columns 7 and 8 as the average fleet size and computational time on 10 independent runs. We compare the performance of our heuristic algorithm to the results obtained with the modified version of the GUTS implementation of [33] (GUTS in columns 9 and 10). Finally, the average percentage gap of the modular heuristic with respect to GUTS, on each problem instance and for each value considered for ϵ, is reported in column 10 (a negative value means a better performance of the modular heuristic). Note that, in each of these problem instances, the minimum and maximum visit frequencies are set to one and the considered number of periods, respectively.

As shown in Tables 6 and 7, the proposed modular heuristic algorithm always produces either equivalent or better results compared to GUTS. However, the average percentage gap between the two algorithms clearly varies depending on the values set as the upper bound of the budget constraint. In the case where the ϵ value is set to ϵ^*, the average percentage gap is -2.47% indicating that the modular heuristic performs significantly better than GUTS. On the other hand, in the cases where the budget constraint is tightened by fixing the ϵ value to $0.8\epsilon^*$ and $0.9\epsilon^*$, both algorithms face a more challenging task to produce good results and, consequently, their performance slightly worsens. However, the results show that the tighter the budget constraint is, the more the modular heuristic shows its superiority to produce better results. These results reveal the fact that the proposed heuristic algorithm has better capability, compared to GUTS, to address fs-PVRP instances, especially for those problems having more restricted search space. The average percentage gaps between the two algorithms are respectively -4.27% and -7.8%, for the $0.8\epsilon^*$ and $0.9\epsilon^*$ cases. Contrary, increasing the upper bounds to $1.1\epsilon^*$ and $1.2\epsilon^*$ results in producing better solutions by both algorithms. In these cases, the average percentage gaps change to -3.87% and -3.87%.

Results on fs-MDVRP instances are displayed in Tables 8 and 9, where, as shown in Tables 6 and 7, columns 2, 3 and 4 represent respectively instance identifier, number of customers, number of

Table 4
Sensitivity analysis with model R_1.

Problem class	EX01	EX02	EX03	EX04	EX05	EX06	EX07
PVRP							
OFV (%)	–	+14.04	–	+7.40	+1.44	–	–
Time (%)	–	−1.70	–	−3.12	−0.72	–	–
MDVRP							
OFV (%)	–	–	+5.22	+8.21	+1.61	+0.90	+0.6
Time (%)	–	–	−1.88	−3.14	−0.70	−0.60	+0.8
MDPVRP							
OFV (%)	+0.8	+21.13	+5.92	+8.92	+1.85	+0.90	+0.9
Time (%)	+1.44	−2.31	−1.96	−3.14	−0.76	−0.80	+1.12

Table 5
Sensitivity analysis with model R_2.

Problem class	EX01	EX02	EX03	EX04	EX05	EX06	EX07
PVRP							
OFV (%)	–	+15.25	–	+7.51	+1.60	–	–
Time (%)	–	−1.82	–	−3.31	−0.80	–	–
MDVRP							
OFV (%)	–	–	+5.87	+8.44	+1.66	+1.21	+0.6
Time (%)	–	–	−1.89	−3.32	−0.82	−0.8	+1.13
MDPVRP							
OFV (%)	+0.90	+22.09	+6.11	+8.96	+1.88	+1.52	+1.33
Time (%)	+1.67	−2.81	−2.21	−3.33	−0.82	−0.9	+1.45

Table 6
Results on fs-PVRP instances with model R_1.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr01	48	1	4	2	$0.8\epsilon^*$	3	0.25	3	0.37	0
					$0.9\epsilon^*$	3	0.22	3	0.35	0
					ϵ^*	2	0.22	2	0.35	0
					$1.1\epsilon^*$	2	0.20	2	0.35	0
					$1.2\epsilon^*$	2	0.16	2	0.33	0
pr02	96	1	4	4	$0.8\epsilon^*$	5	0.57	5	0.94	0
					$0.9\epsilon^*$	5	0.51	5	0.85	0
					ϵ^*	4	0.50	4	0.80	0
					$1.1\epsilon^*$	4	0.46	4	0.75	0
					$1.2\epsilon^*$	4	0.42	4	0.71	0
pr03	144	1	4	6	$0.8\epsilon^*$	7	4.91	7	6.69	0
					$0.9\epsilon^*$	6	4.66	7	6.13	−14
					ϵ^*	6	3.95	6	5.56	0
					$1.1\epsilon^*$	5	3.44	6	5.38	−17
					$1.2\epsilon^*$	5	3.31	6	5.09	−17
pr04	192	1	4	8	$0.8\epsilon^*$	9	8.52	10	9.57	−10
					$0.9\epsilon^*$	9	8.29	9	9.12	0
					ϵ^*	7	7.48	8	8.19	−13
					$1.1\epsilon^*$	7	7.24	8	8.10	−13
					$1.2\epsilon^*$	7	6.75	8	7.92	−13
pr05	240	1	4	10	$0.8\epsilon^*$	12	16.05	14	19.91	−14
					$0.9\epsilon^*$	11	15.92	14	19.72	−21
					ϵ^*	10	14.92	10	18.22	0
					$1.1\epsilon^*$	9	14.33	10	17.71	−10
					$1.2\epsilon^*$	9	13.79	10	17.24	−10
pr06	288	1	4	12	$0.8\epsilon^*$	14	16.50	16	19.36	−13
					$0.9\epsilon^*$	13	16.16	16	19.29	−19
					ϵ^*	12	15.72	14	19.11	−14
					$1.1\epsilon^*$	12	15.37	13	18.72	−8
					$1.2\epsilon^*$	12	15.30	13	18.33	−8
pr07	126	1	5	4	$0.8\epsilon^*$	5	4.82	5	5.46	0
					$0.9\epsilon^*$	4	4.51	5	5.28	−20
					ϵ^*	4	4.30	4	5.09	0
					$1.1\epsilon^*$	4	4.16	4	4.91	0
					$1.2\epsilon^*$	4	4.04	4	4.77	0
pr08	72	1	6	3	$0.8\epsilon^*$	4	0.39	4	0.89	0
					$0.9\epsilon^*$	4	0.35	4	0.85	0
					ϵ^*	3	0.30	3	0.80	0

Table 6 (*continued*)

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
					$1.1\epsilon^*$	3	0.28	3	0.75	0
					$1.2\epsilon^*$	3	0.28	3	0.74	0
pr09	144	1	6	6	$0.8\epsilon^*$	7	5.64	8	7.50	−13
					$0.9\epsilon^*$	7	5.12	7	7.39	0
					ϵ^*	6	4.76	6	7.23	0
					$1.1\epsilon^*$	6	4.55	6	7.14	0
					$1.2\epsilon^*$	6	4.31	6	7.11	0
pr10	216	1	6	9	$0.8\epsilon^*$	11	17.40	11	18.99	0
					$0.9\epsilon^*$	10	17.02	11	18.96	−10
					ϵ^*	10	16.85	11	18.89	−10
					$1.1\epsilon^*$	9	16.70	9	18.69	0
					$1.2\epsilon^*$	9	16.50	9	18.44	0
pr11	288	1	6	12	$0.8\epsilon^*$	14	19.14	16	19.54	−13
					$0.9\epsilon^*$	13	19.07	16	19.23	−19
					ϵ^*	12	18.94	12	19.01	0
					$1.1\epsilon^*$	12	18.80	12	18.86	0
					$1.2\epsilon^*$	12	18.56	12	18.77	0
pr12	417	1	7	9	$0.8\epsilon^*$	12	20	14	20	−14
					$0.9\epsilon^*$	12	20	14	20	−14
					ϵ^*	9	20	10	20	−10
					$1.1\epsilon^*$	9	20	10	20	−10
					$1.2\epsilon^*$	9	20	10	20	−10

Table 7
Results on fs-PVRP instances with model R_1 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr13	100	1	8	4	$0.8\epsilon^*$	5	0.61	5	0.99	0
					$0.9\epsilon^*$	5	0.53	5	0.91	0
					ϵ^*	4	0.50	4	0.86	0
					$1.1\epsilon^*$	4	0.46	4	0.80	0
					$1.2\epsilon^*$	4	0.41	4	0.75	0
pr14	75	1	10	1	$0.8\epsilon^*$	2	0.38	2	0.69	0
					$0.9\epsilon^*$	1	0.35	1	0.62	0
					ϵ^*	1	0.32	1	0.59	0
					$1.1\epsilon^*$	1	0.30	1	0.54	0
					$1.2\epsilon^*$	1	0.30	1	0.50	0
pr15	75	1	10	1	$0.8\epsilon^*$	2	0.44	2	0.71	0
					$0.9\epsilon^*$	2	0.40	2	0.65	0
					ϵ^*	1	0.38	1	0.63	0
					$1.1\epsilon^*$	1	0.35	1	0.60	0
					$1.2\epsilon^*$	1	0.34	1	0.57	0

depots and number of periods. The results obtained by the heuristic algorithm are compared, once again, to the modified version of GUTS implemented in [33], in terms of solution quality and computational time.

The main conclusions derived from Tables 8 and 9 are similar to those stated above for the fs-PVRP. The modular heuristic clearly outperforms GUTS on the majority of problem instances.

Results on fs-MDPVRP instances are finally summarized in Tables 10 and 11 whose structure is similar to that of previous tables. Once again, the results produced by the heuristic algorithm are compared to GUTS in order to assess the efficiency of the algorithm, in terms of solution quality and computational time.

Tables 10 and 11 show that the proposed modular heuristic algorithm is a competitive solution methodology, able to produce high-quality solutions in a reasonable time. As for the two previous problems, the relative performance of two algorithms,

Table 8
Results on fs-MDVRP instances with model R_1.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr01	48	4	1	4	$0.8\epsilon^*$	5	0.15	5	0.15	0
					$0.9\epsilon^*$	5	0.12	5	0.15	0
					ϵ^*	4	0.10	4	0.14	0
					$1.1\epsilon^*$	4	0.08	4	0.14	0
					$1.2\epsilon^*$	4	0.08	4	0.13	0
pr02	96	4	1	8	$0.8\epsilon^*$	8	0.41	9	0.81	−11
					$0.9\epsilon^*$	8	0.38	9	0.74	−11
					ϵ^*	7	0.34	8	0.69	−13
					$1.1\epsilon^*$	7	0.32	8	0.62	−13
					$1.2\epsilon^*$	6	0.30	8	0.60	−25
pr03	144	4	1	12	$0.8\epsilon^*$	13	1.14	14	1.52	−7
					$0.9\epsilon^*$	12	1.09	13	1.44	−7
					ϵ^*	12	0.68	12	1.39	0
					$1.1\epsilon^*$	12	0.62	12	1.30	0
					$1.2\epsilon^*$	12	0.57	12	1.26	0
pr04	192	4	1	16	$0.8\epsilon^*$	16	3.95	17	4.68	−6
					$0.9\epsilon^*$	16	3.82	17	4.60	−6
					ϵ^*	15	3.79	16	4.51	−6
					$1.1\epsilon^*$	15	3.74	16	4.46	−6
					$1.2\epsilon^*$	15	3.69	16	4.39	−6
pr05	240	4	1	20	$0.8\epsilon^*$	21	7.77	22	9.51	−5
					$0.9\epsilon^*$	20	7.66	22	9.32	−9
					ϵ^*	20	7.41	20	9.16	0
					$1.1\epsilon^*$	20	7.38	20	9.09	0
					$1.2\epsilon^*$	20	7.25	20	9.03	0
pr06	288	4	1	24	$0.8\epsilon^*$	25	8.99	27	9.62	−7
					$0.9\epsilon^*$	25	8.92	27	9.56	−7
					ϵ^*	24	8.87	24	9.44	0
					$1.1\epsilon^*$	24	8.74	24	9.37	0
					$1.2\epsilon^*$	24	8.69	24	9.28	0
pr07	72	6	1	6	$0.8\epsilon^*$	7	0.38	7	0.55	0
					$0.9\epsilon^*$	6	0.32	7	0.54	−14
					ϵ^*	6	0.29	6	0.51	0
					$1.1\epsilon^*$	6	0.26	6	0.47	0
					$1.2\epsilon^*$	6	0.25	6	0.41	0
pr08	144	6	1	12	$0.8\epsilon^*$	13	1.75	13	1.97	0
					$0.9\epsilon^*$	13	1.70	13	1.93	0
					ϵ^*	12	1.62	12	1.89	0
					$1.1\epsilon^*$	12	1.56	12	1.83	0
					$1.2\epsilon^*$	11	1.50	12	1.79	−8

Table 8 (*continued*)

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr09	216	6	1	18	$0.8\epsilon^*$	20	7.91	22	8.84	−9
					$0.9\epsilon^*$	20	7.84	22	8.78	−9
					ϵ^*	18	7.72	19	8.71	−5
					$1.1\epsilon^*$	18	7.64	19	8.64	−5
					$1.2\epsilon^*$	18	7.59	19	8.59	−5
pr10	288	6	1	24	$0.8\epsilon^*$	25	8.84	26	9.73	−4
					$0.9\epsilon^*$	24	8.76	26	9.64	−8
					ϵ^*	24	8.67	24	9.57	0
					$1.1\epsilon^*$	24	8.62	24	9.51	0
					$1.2\epsilon^*$	24	8.58	24	9.46	0
pr11	240	6	1	10	$0.8\epsilon^*$	12	7.79	14	9.21	−14
					$0.9\epsilon^*$	12	7.70	14	9.09	−14
					ϵ^*	10	7.59	11	8.89	−9
					$1.1\epsilon^*$	10	7.50	10	8.80	0
					$1.2\epsilon^*$	10	7.39	10	8.69	0
pr12	240	6	1	10	$0.8\epsilon^*$	13	7.91	15	9.45	−13
					$0.9\epsilon^*$	12	7.80	14	9.40	−14
					ϵ^*	10	7.66	11	9.32	−9
					$1.1\epsilon^*$	10	7.60	10	9.28	0
					$1.2\epsilon^*$	10	7.53	10	9.21	0

Table 9
Results on fs-MDVRP instances with model R_1 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr13	360	9	1	10	$0.8\epsilon^*$	13	10.80	16	14.73	−18
					$0.9\epsilon^*$	11	10.41	13	14.39	−15
					ϵ^*	10	10.09	11	14.07	−9
					$1.1\epsilon^*$	10	9.83	10	13.88	0
					$1.2\epsilon^*$	10	9.66	10	13.53	0
pr14	360	9	1	10	$0.8\epsilon^*$	12	9.82	14	14.66	−14
					$0.9\epsilon^*$	10	9.64	13	14.59	−23
					ϵ^*	10	9.32	10	14.21	0
					$1.1\epsilon^*$	10	9.18	10	14.06	0
					$1.2\epsilon^*$	10	9.02	10	14.04	0
pr15	360	9	1	10	$0.8\epsilon^*$	12	10.44	14	14.54	−14
					$0.9\epsilon^*$	12	10.23	12	14.21	0
					ϵ^*	10	10.10	10	13.99	0
					$1.1\epsilon^*$	10	9.93	10	13.64	0
					$1.2\epsilon^*$	9	9.77	10	13.42	−10

Table 10
Results on fs-MDPVRP instances with model R_1.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr01	48	4	4	4	$0.8\epsilon^*$	5	0.35	5	0.43	0
					$0.9\epsilon^*$	5	0.30	5	0.42	0
					ϵ^*	4	0.28	4	0.40	0
					$1.1\epsilon^*$	4	0.26	4	0.38	0
					$1.2\epsilon^*$	4	0.20	4	0.36	0
pr02	96	4	4	4	$0.8\epsilon^*$	5	0.92	5	1.39	0
					$0.9\epsilon^*$	4	0.85	5	1.34	−20
					ϵ^*	4	0.81	4	1.28	0
					$1.1\epsilon^*$	4	0.72	4	1.26	0
					$1.2\epsilon^*$	4	0.68	4	1.19	0
pr03	144	4	4	8	$0.8\epsilon^*$	9	4.95	9	6.41	0
					$0.9\epsilon^*$	9	4.86	9	6.33	0
					ϵ^*	8	4.80	8	6.23	0
					$1.1\epsilon^*$	8	4.75	8	6.13	0
					$1.2\epsilon^*$	7	4.70	8	6.09	−13
pr04	192	4	4	8	$0.8\epsilon^*$	9	13.83	10	15.66	−10
					$0.9\epsilon^*$	8	13.75	9	15.58	−11
					ϵ^*	7	13.68	8	15.52	−13
					$1.1\epsilon^*$	7	13.62	8	15.44	−13
					$1.2\epsilon^*$	7	13.56	8	15.37	−13
pr05	240	4	4	12	$0.8\epsilon^*$	13	20.88	14	24.33	−7
					$0.9\epsilon^*$	12	20.80	14	24.17	−7
					ϵ^*	12	20.71	12	24.09	0
					$1.1\epsilon^*$	12	20.59	12	23.88	0
					$1.2\epsilon^*$	12	20.39	12	23.82	0
pr06	288	4	4	12	$0.8\epsilon^*$	14	17.55	16	21.45	−13
					$0.9\epsilon^*$	13	17.47	15	21.34	−13
					ϵ^*	12	17.40	14	21.25	−14
					$1.1\epsilon^*$	12	17.36	13	21.20	−8
					$1.2\epsilon^*$	12	17.19	13	21.11	−8
pr07	72	6	6	6	$0.8\epsilon^*$	7	1.52	7	2.06	0
					$0.9\epsilon^*$	7	1.44	7	1.98	0
					ϵ^*	6	1.40	6	1.96	0
					$1.1\epsilon^*$	6	1.33	6	1.91	0
					$1.2\epsilon^*$	6	1.26	6	1.84	0
pr08	144	6	6	6	$0.8\epsilon^*$	7	5.31	8	8.04	−13
					$0.9\epsilon^*$	7	5.27	7	7.97	0
					ϵ^*	6	5.11	6	7.91	0
					$1.1\epsilon^*$	6	5.09	6	7.88	0
					$1.2\epsilon^*$	6	5.01	6	7.82	0
pr09	216	6	6	12	$0.8\epsilon^*$	13	20.77	13	24.19	0
					$0.9\epsilon^*$	12	20.63	14	24.12	−14
					ϵ^*	12	20.44	13	23.96	−8
					$1.1\epsilon^*$	12	20.21	13	23.89	−8
					$1.2\epsilon^*$	12	19.92	13	23.81	−8
pr10	288	6	6	18	$0.8\epsilon^*$	19	23.31	20	26.79	−5
					$0.9\epsilon^*$	19	23.19	20	26.71	−5
					ϵ^*	18	22.95	18	26.53	0
					$1.1\epsilon^*$	18	22.88	18	26.49	0
					$1.2\epsilon^*$	18	22.80	18	26.41	0
pr11	50	4	2	8	$0.8\epsilon^*$	9	0.48	10	0.88	−10
					$0.9\epsilon^*$	8	0.45	8	0.85	0
					ϵ^*	8	0.42	8	0.80	0
					$1.1\epsilon^*$	7	0.40	7	0.76	0
					$1.2\epsilon^*$	7	0.40	7	0.72	0
pr12	50	2	5	6	$0.8\epsilon^*$	8	0.61	8	0.97	0
					$0.9\epsilon^*$	8	0.57	8	0.95	0
					ϵ^*	6	0.52	6	0.93	0
					$1.1\epsilon^*$	5	0.48	5	0.90	0
					$1.2\epsilon^*$	5	0.48	5	0.89	0

Table 11
Results on fs-MDPVRP instances with model R_1 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr13	75	2	5	6	$0.8\epsilon^*$	7	0.52	9	1.01	−22
					$0.9\epsilon^*$	6	0.45	8	0.95	−25
					ϵ^*	6	0.42	6	0.88	0
					$1.1\epsilon^*$	6	0.40	6	0.81	0
					$1.2\epsilon^*$	5	0.35	5	0.75	0
pr14	100	3	2	10	$0.8\epsilon^*$	11	0.99	11	1.43	0
					$0.9\epsilon^*$	10	0.95	11	1.40	−9
					ϵ^*	10	0.90	10	1.35	0
					$1.1\epsilon^*$	10	0.86	10	1.30	0
					$1.2\epsilon^*$	10	0.82	10	1.27	0
pr15	100	3	5	12	$0.8\epsilon^*$	14	0.99	16	1.56	−12

Table 11 (*continued*)

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.) K	Heuristic (ave.) Time (min)	GUTS (ave.) K	GUTS (ave.) Time (min)	Gap (%)
					$0.9\epsilon^*$	14	0.95	16	1.50	−12
					ϵ^*	12	0.88	13	1.46	−7
					$1.1\epsilon^*$	12	0.82	12	1.40	0
					$1.2\epsilon^*$	12	0.77	12	1.38	0

Table 12
ASD per instance on instances with model R_1.

ϵ	MHA (%) fs-PVRP	MHA (%) fs-MDVRP	MHA (%) fs-MDPVRP	GUTS (%) fs-PVRP	GUTS (%) fs-MDVRP	GUTS (%) fs-MDPVRP
$0.8\epsilon^*$	0.18	0.09	0.23	0.24	0.14	0.30
$0.9\epsilon^*$	0.18	0.09	0.23	0.22	0.12	0.30
ϵ^*	0.15	0.08	0.21	0.18	0.12	0.30
$1.1\epsilon^*$	0.14	0.05	0.20	0.18	0.08	0.25
$1.2\epsilon^*$	0.12	0.05	0.20	0.16	0.08	0.24

Table 13
Results on fs-PVRP instances with model R_2.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.) K	Heuristic (ave.) Time (min)	GUTS (ave.) K	GUTS (ave.) Time (min)	Gap (%)
pr01	48	4	4	2	$0.8\epsilon^*$	3	0.22	3	0.39	0
					$0.9\epsilon^*$	3	0.20	3	0.36	0
					ϵ^*	2	0.20	2	0.35	0
					$1.1\epsilon^*$	2	0.18	2	0.34	0
					$1.2\epsilon^*$	2	0.18	2	0.33	0
pr02	96	4	4	4	$0.8\epsilon^*$	5	0.57	5	0.94	0
					$0.9\epsilon^*$	4	0.51	5	0.85	−20
					ϵ^*	4	0.49	4	0.82	0
					$1.1\epsilon^*$	4	0.47	4	0.77	0
					$1.2\epsilon^*$	4	0.44	4	0.73	0
pr03	144	4	4	6	$0.8\epsilon^*$	7	4.20	8	6.38	−13
					$0.9\epsilon^*$	7	4.09	8	6.16	−13
					ϵ^*	6	3.92	6	5.59	0
					$1.1\epsilon^*$	6	3.80	6	5.48	0
					$1.2\epsilon^*$	6	3.66	6	5.32	0
pr04	192	4	4	8	$0.8\epsilon^*$	9	8.60	10	9.61	−10
					$0.9\epsilon^*$	8	8.54	10	9.24	−20
					ϵ^*	8	7.55	8	8.25	0
					$1.1\epsilon^*$	7	7.36	8	8.19	−13
					$1.2\epsilon^*$	7	7.28	8	7.95	−13
pr05	240	4	4	10	$0.8\epsilon^*$	12	16.14	14	20.19	−14
					$0.9\epsilon^*$	11	15.85	14	18.95	−21
					ϵ^*	10	15.21	11	18.39	−9
					$1.1\epsilon^*$	10	14.93	11	18.11	−9
					$1.2\epsilon^*$	10	14.78	11	17.78	−9
pr06	288	4	4	12	$0.8\epsilon^*$	14	16.72	16	19.50	−13
					$0.9\epsilon^*$	14	16.45	16	19.42	−13
					ϵ^*	13	15.61	14	19.32	−7
					$1.1\epsilon^*$	13	16.21	14	18.80	−7
					$1.2\epsilon^*$	12	15.99	13	18.53	−7
pr07	126	1	5	4	$0.8\epsilon^*$	5	4.65	5	6.72	0
					$0.9\epsilon^*$	4	4.55	5	6.42	−20
					ϵ^*	4	4.36	4	6.07	0
					$1.1\epsilon^*$	4	3.30	4	5.78	0
					$1.2\epsilon^*$	4	3.19	4	5.39	0
pr08	72	6	6	3	$0.8\epsilon^*$	4	0.44	4	0.93	0
					$0.9\epsilon^*$	3	0.42	4	0.87	−0.25
					ϵ^*	3	0.40	3	0.82	0
					$1.1\epsilon^*$	3	0.36	3	0.75	0
					$1.2\epsilon^*$	3	0.34	3	0.70	0
pr09	144	6	6	6	$0.8\epsilon^*$	7	5.85	8	7.53	−13
					$0.9\epsilon^*$	7	5.31	7	7.44	0
					ϵ^*	6	4.94	6	7.39	0
					$1.1\epsilon^*$	6	4.77	6	7.22	0
					$1.2\epsilon^*$	6	4.60	6	7.16	0
pr10	216	6	6	9	$0.8\epsilon^*$	10	17.30	12	19.12	−17
					$0.9\epsilon^*$	10	16.85	12	19.04	−17
					ϵ^*	10	16.39	11	18.96	−10
					$1.1\epsilon^*$	9	16.27	9	18.78	−10
					$1.2\epsilon^*$	9	16.19	9	18.53	−10
pr11	288	6	6	12	$0.8\epsilon^*$	14	19.16	16	19.61	−13
					$0.9\epsilon^*$	14	19.05	16	19.44	−13
					ϵ^*	12	18.99	13	19.10	−8
					$1.1\epsilon^*$	12	18.87	13	18.97	−8
					$1.2\epsilon^*$	12	18.66	13	18.86	−8
pr12	417	1	7	9	$0.8\epsilon^*$	12	20	14	20	−14
					$0.9\epsilon^*$	12	20	14	20	−14
					ϵ^*	10	20	10	20	0
					$1.1\epsilon^*$	9	20	10	20	−10
					$1.2\epsilon^*$	9	20	10	20	−10

Table 14
Results on fs-PVRP instances with model R_2 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.) K	Heuristic (ave.) Time (min)	GUTS (ave.) K	GUTS (ave.) Time (min)	Gap (%)
pr13	100	1	8	1	$0.8\epsilon^*$	5	0.52	5	0.94	0
					$0.9\epsilon^*$	5	0.48	5	0.85	0
					ϵ^*	4	0.46	4	0.80	0
					$1.1\epsilon^*$	4	0.42	4	0.75	0
					$1.2\epsilon^*$	4	0.39	4	0.71	0
pr14	75	1	10	1	$0.8\epsilon^*$	5	0.52	5	0.94	0
					$0.9\epsilon^*$	5	0.48	5	0.85	0
					ϵ^*	4	0.46	4	0.80	0
					$1.1\epsilon^*$	4	0.42	4	0.75	0
					$1.2\epsilon^*$	4	0.39	4	0.71	0
pr15	75	1	10	1	$0.8\epsilon^*$	5	0.52	5	0.94	0
					$0.9\epsilon^*$	5	0.48	5	0.85	0
					ϵ^*	4	0.46	4	0.80	0
					$1.1\epsilon^*$	4	0.42	4	0.75	0
					$1.2\epsilon^*$	4	0.39	4	0.71	0

measured by the average percentage gap, depends upon the on values assigned to ϵ.

At the end of this section, it is worth focusing on the Average Standard Deviation (ASD) per instance of the algorithms in each considered problem class. Table 12 summarizes the ASD per instance values obtained by MHA and GUTS on the fs-PVRP, fs-MDVRP and fs-MDPVRP problem instances. The results shown by this table reveal the fact that both algorithm are very reliable.

5.3.2. Model R_2

In model R_2, we investigate how the algorithm reacts when an upper bound is imposed on the total distance that all vehicles assigned to each depot are permitted to travel in each period. The specifications of how the algorithms are applied in the case of model R_2 are exactly the same as in the case of model R_1 (i.e., the same number of runs is applied as well as identical maximum allotted run-times for the algorithms are considered for each problem).

Tables 13–18 display the results obtained by the heuristic algorithm on the fs-PVRP, fs-MDVRP and fs-MDPVRP problems. Each of these tables use the same structure considered for the tables of the previous subsection. It should be noted that, in these tables, ϵ^* is a $T \times D$ matrix, in which ϵ_{td} corresponds to the total

Table 15
Results on fs-MDVRP instances with model R_2.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (Ave.)		GUTS (Ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr01	48	4	4	4	$0.8\epsilon^*$	5	0.14	5	0.17	0
					$0.9\epsilon^*$	5	0.12	5	0.17	0
					ϵ^*	4	0.10	4	0.16	0
					$1.1\epsilon^*$	4	0.09	4	0.14	0
					$1.2\epsilon^*$	4	0.07	4	0.14	0
pr02	96	4	4	8	$0.8\epsilon^*$	9	0.43	10	0.84	0
					$0.9\epsilon^*$	9	0.36	9	0.77	−10
					ϵ^*	8	0.33	8	0.72	0
					$1.1\epsilon^*$	8	0.31	8	0.66	0
					$1.2\epsilon^*$	8	0.30	8	0.62	0
pr03	144	4	4	12	$0.8\epsilon^*$	13	1.08	14	1.55	−7
					$0.9\epsilon^*$	13	1.04	13	1.45	0
					ϵ^*	12	0.59	12	1.44	0
					$1.1\epsilon^*$	12	0.59	12	1.35	0
					$1.2\epsilon^*$	12	0.48	12	1.29	0
pr04	192	4	4	16	$0.8\epsilon^*$	17	3.90	18	4.72	−6
					$0.9\epsilon^*$	17	3.85	18	4.66	−6
					ϵ^*	16	3.80	17	4.60	−6
					$1.1\epsilon^*$	16	3.74	17	4.52	−6
					$1.2\epsilon^*$	16	3.70	17	4.47	−6
pr05	240	4	4	20	$0.8\epsilon^*$	21	7.53	22	9.59	−5
					$0.9\epsilon^*$	21	7.49	22	9.35	−5
					ϵ^*	20	7.34	21	9.33	−5
					$1.1\epsilon^*$	20	7.29	21	9.19	−5
					$1.2\epsilon^*$	20	7.22	21	9.14	−5
pr06	288	4	4	24	$0.8\epsilon^*$	25	9.01	27	9.84	−7
					$0.9\epsilon^*$	25	8.97	27	9.60	−7
					ϵ^*	24	8.80	25	9.49	−4
					$1.1\epsilon^*$	24	8.75	25	9.40	−4
					$1.2\epsilon^*$	24	8.69	25	9.33	−4
pr07	72	6	6	6	$0.8\epsilon^*$	7	0.3	7	0.64	0
					$0.9\epsilon^*$	7	0.29	7	0.59	0
					ϵ^*	6	0.27	6	0.54	0
					$1.1\epsilon^*$	6	0.25	6	0.49	0
					$1.2\epsilon^*$	6	0.24	6	0.46	0
pr08	144	6	6	12	$0.8\epsilon^*$	13	1.73	13	2.04	0
					$0.9\epsilon^*$	13	1.69	13	1.99	0
					ϵ^*	12	1.62	12	1.95	0
					$1.1\epsilon^*$	12	1.56	12	1.88	0
					$1.2\epsilon^*$	12	1.50	12	1.84	0
pr09	216	6	6	18	$0.8\epsilon^*$	20	7.95	22	8.91	−9
					$0.9\epsilon^*$	21	7.84	22	8.84	−4
					ϵ^*	18	7.79	21	8.79	−14
					$1.1\epsilon^*$	18	7.71	21	8.74	−10
					$1.2\epsilon^*$	18	7.64	21	8.63	−10
pr10	288	6	6	24	$0.8\epsilon^*$	25	8.99	26	9.82	−4
					$0.9\epsilon^*$	25	8.92	26	9.72	−4
					ϵ^*	24	8.85	25	9.65	−4
					$1.1\epsilon^*$	24	8.73	25	9.55	−4
					$1.2\epsilon^*$	24	8.62	25	9.48	−4
pr11	240	6	1	10	$0.8\epsilon^*$	12	7.95	14	9.44	−14
					$0.9\epsilon^*$	12	7.90	14	9.40	−14
					ϵ^*	10	7.84	10	9.29	0
					$1.1\epsilon^*$	10	7.66	10	9.06	0
					$1.2\epsilon^*$	10	7.60	10	8.89	0
pr12	240	6	1	10	$0.8\epsilon^*$	13	7.91	15	9.60	−13
					$0.9\epsilon^*$	13	7.88	14	9.52	−7
					ϵ^*	10	7.81	11	9.49	−9
					$1.1\epsilon^*$	10	7.66	10	9.38	0
					$1.2\epsilon^*$	10	7.60	10	9.30	0

Table 16
Results on fs-MDVRP instances with model R_2 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (Ave.)		GUTS (Ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr13	360	9	1	10	$0.8\epsilon^*$	14	11.12	16	14.88	−12
					$0.9\epsilon^*$	11	10.41	13	14.80	−15
					ϵ^*	10	10.09	11	14.67	−9
					$1.1\epsilon^*$	10	9.83	10	14.34	0
					$1.2\epsilon^*$	10	9.66	10	14.09	0
pr14	360	9	1	10	$0.8\epsilon^*$	12	9.82	15	14.75	−20
					$0.9\epsilon^*$	11	9.74	13	14.70	−15
					ϵ^*	10	9.55	10	14.39	0
					$1.1\epsilon^*$	10	9.34	10	14.16	0
					$1.2\epsilon^*$	10	9.19	10	14.10	0
pr15	360	9	1	10	$0.8\epsilon^*$	13	10.71	15	14.60	−13
					$0.9\epsilon^*$	12	10.37	12	14.44	0
					ϵ^*	10	10.21	10	14.19	0
					$1.1\epsilon^*$	10	10.04	10	13.87	0
					$1.2\epsilon^*$	10	9.85	10	13.56	0

Table 17
Results on fs−MDPVRP instances with model R_2.

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr01	48	4	4	4	$0.8\epsilon^*$	5	0.50	5	0.50	0
					$0.9\epsilon^*$	5	0.42	5	0.47	0
					ϵ^*	4	0.36	4	0.44	0
					$1.1\epsilon^*$	4	0.30	4	0.41	0
					$1.2\epsilon^*$	4	0.30	4	0.37	0
pr02	96	4	4	4	$0.8\epsilon^*$	5	0.97	5	1.43	0
					$0.9\epsilon^*$	5	0.92	5	1.38	0
					ϵ^*	4	0.87	4	1.31	0
					$1.1\epsilon^*$	4	0.80	4	1.28	0
					$1.2\epsilon^*$	4	0.79	4	1.22	0
pr03	144	4	4	8	$0.8\epsilon^*$	9	4.99	10	6.46	0
					$0.9\epsilon^*$	8	4.96	10	6.41	−20
					ϵ^*	8	4.90	9	6.29	−11
					$1.1\epsilon^*$	8	4.83	9	6.18	−11
					$1.2\epsilon^*$	8	4.78	9	6.11	−11
pr04	192	4	4	8	$0.8\epsilon^*$	9	13.95	9	15.70	0
					$0.9\epsilon^*$	8	13.80	9	15.66	−11
					ϵ^*	8	13.71	8	15.64	0
					$1.1\epsilon^*$	7	13.66	8	15.49	−13
					$1.2\epsilon^*$	7	13.60	8	15.53	−13
pr05	240	4	4	12	$0.8\epsilon^*$	13	20.91	14	24.55	−7
					$0.9\epsilon^*$	12	20.77	14	24.48	−14
					ϵ^*	12	20.61	13	24.33	−8
					$1.1\epsilon^*$	12	20.57	13	23.94	−8
					$1.2\epsilon^*$	12	20.50	13	23.85	−8
pr06	288	4	4	12	$0.8\epsilon^*$	14	17.85	16	21.51	−13
					$0.9\epsilon^*$	14	17.66	16	21.40	−13
					ϵ^*	12	17.50	14	21.34	−14
					$1.1\epsilon^*$	12	17.46	14	21.29	−14
					$1.2\epsilon^*$	12	17.40	14	21.20	−14
pr07	72	6	6	6	$0.8\epsilon^*$	7	1.59	7	2.09	0
					$0.9\epsilon^*$	7	1.54	7	2.06	0
					ϵ^*	6	1.48	6	2.01	0
					$1.1\epsilon^*$	6	1.42	6	1.97	0
					$1.2\epsilon^*$	6	1.41	6	1.89	0
pr08	144	6	6	6	$0.8\epsilon^*$	7	5.70	8	8.09	−13
					$0.9\epsilon^*$	7	5.59	7	8.04	0
					ϵ^*	6	5.44	6	7.95	0
					$1.1\epsilon^*$	6	5.31	6	7.94	0
					$1.2\epsilon^*$	5	5.20	6	7.88	−17
pr09	216	6	6	12	$0.8\epsilon^*$	13	20.70	15	24.22	−7
					$0.9\epsilon^*$	13	20.51	15	24.17	−7
					ϵ^*	12	20.42	13	24.12	−7
					$1.1\epsilon^*$	12	20.35	13	23.95	−7

Table 17 (*continued*)

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
					$1.2\epsilon^*$	12	20.27	13	23.87	−7
pr10	288	6	6	18	$0.8\epsilon^*$	19	23.32	20	26.86	−5
					$0.9\epsilon^*$	18	23.25	19	26.76	−5
					ϵ^*	18	23.04	18	26.69	0
					$1.1\epsilon^*$	18	22.88	18	26.57	0
					$1.2\epsilon^*$	18	22.85	18	26.46	0
pr11	50	4	2	8	$0.8\epsilon^*$	9	0.51	10	0.96	−10
					$0.9\epsilon^*$	8	0.50	9	0.90	−11
					ϵ^*	8	0.46	8	0.84	0
					$1.1\epsilon^*$	7	0.44	7	0.80	0
					$1.2\epsilon^*$	7	0.42	7	0.79	0
pr12	50	2	5	6	$0.8\epsilon^*$	8	0.66	8	1.09	0
					$0.9\epsilon^*$	8	0.60	8	0.99	0
					ϵ^*	6	0.56	6	0.95	0
					$1.1\epsilon^*$	5	0.51	5	0.94	0
					$1.2\epsilon^*$	5	0.51	5	0.93	0

Table 18
Results on fs-MDPVRP instances with model R_2 (cont.).

Instance	N	M	T	Vehicle-BKS	ϵ	Heuristic (ave.)		GUTS (ave.)		Gap (%)
						K	Time (min)	K	Time (min)	
pr13	75	2	5	6	$0.8\epsilon^*$	8	0.60	9	1.22	−11
					$0.9\epsilon^*$	6	0.52	8	1.09	−25
					ϵ^*	6	0.46	6	0.96	0
					$1.1\epsilon^*$	6	0.44	6	0.90	0
					$1.2\epsilon^*$	5	0.40	5	0.84	0
pr14	100	3	2	10	$0.8\epsilon^*$	11	1.20	11	1.43	0
					$0.9\epsilon^*$	10	1.07	11	1.59	−9
					ϵ^*	10	0.99	10	1.47	0
					$1.1\epsilon^*$	10	0.88	10	1.34	0
					$1.2\epsilon^*$	10	0.88	10	1.32	0
pr15	100	3	5	12	$0.8\epsilon^*$	15	1.15	16	1.77	−6
					$0.9\epsilon^*$	14	1.11	16	1.63	−12
					ϵ^*	12	1.03	13	1.50	−7
					$1.1\epsilon^*$	12	0.93	12	1.44	0
					$1.2\epsilon^*$	12	0.82	12	1.39	0

Table 19
ASD per instance on instances with model R_2.

ϵ	MHA (%)			GUTS (%)		
	fs-PVRP	fs-MDVRP	fs-MDPVRP	fs-PVRP	fs-MDVRP	fs-MDPVRP
$0.8\epsilon^*$	0.28	0.16	0.32	0.38	0.22	0.39
$0.9\epsilon^*$	0.27	0.16	0.30	0.37	0.22	0.37
ϵ^*	0.22	0.14	0.30	0.35	0.21	0.37
$1.1\epsilon^*$	0.21	0.14	0.28	0.30	0.19	0.33
$1.2\epsilon^*$	0.20	0.14	0.28	0.29	0.19	0.33

distance traveled by vehicles assigned to depot d in period t of the BKS. The performance of the heuristic algorithm is compared, on each set of test problems, to the modified version of the GUTS implementation of [33].

As shown in Tables 13–18, the proposed modular heuristic algorithm performs impressively well relative to the GUTS, in terms of solution quality and computational time.

Table 19 presents the ASD per instance values produced by MHA and GUTS on the fs-PVRP, fs-MDVRP and fs-MDPVRP problem instances.

Studying the results presented by Table 19, along with those of Tables 13–18, shows that restricting the search space by considering model R_2 may result in a decrease in the quality of the considered algorithms. However, both algorithms still perform very well in producing good results, in terms of solution quality and computational efficiency.

6. Conclusions

This paper presented a new modular heuristic algorithm for addressing the fleet-sizing for the multi-depot and periodic vehicle routing problem, setting under vehicle capacity, route duration and budget constraints.

This paper introduced several methodological contributions, particularly, a self-learning mechanism that leads the algorithm to assign better visit patterns to customers, and also to assign customers to better depots as the solution process evolves. This learning mechanism, in addition to other components of the algorithm, provided the capability of the heuristic algorithm to reach high-quality solutions.

To validate the efficiency of the proposed heuristic algorithm, different instances, existing in the literature, were addressed. The computational results showed that the proposed algorithm performs very well.

Acknowledgements

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs, by the Fonds de Recherche du Québec – Nature et Technologies (FRQ-NT) through its Team Research Project program, and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants.

References

[1] Dantzig GB, Ramser JH. The truck dispatching problem. Manag Sci 1959;6:80–91.
[2] Parthanadee P, Logendran R. Periodic product distribution from multi-depots under limited supplies. IIE Trans 2006;38:1009–26.
[3] Hadjiconstantinou E, Baldacci R. A multi-depot period vehicle routing problem arising in the utilities sector. J Oper Res Soc 1998;49:1239–48.
[4] Alegre J, Laguna M, Pacheco J. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. Eur J Oper Res 2007;179(3):736–46.
[5] Cordeau J, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30:105–19.
[6] Vidal T, Crainic T, Gendreau M, Lahrichi N, Rei W. A hybrid genetic algorithm for multi-depots and periodic vehicle routing problems. Oper Res 2012;60:611–24.
[7] Tillman FA. The multiple terminal delivery problem with probabilistic demands. Transp Sci 1969;3:192–204.
[8] Tillman FA, Hering R. A study of a look-ahead procedure for solving the multiterminal delivery problem. Transp Res 1971;5:225–9.
[9] Tillman FA, Cain T. An upper bound algorithm for the single and multiple terminal delivery problem. Manag Sci 1972;18:664–82.
[10] Golden BL, Magnanti TL, Nguyen HQ. Implementing vehicle routing algorithms. Networks 1977;7:113–48.
[11] Raft OM. A modular algorithm for an extended vehicle scheduling problem. Eur J Oper Res 1982;11:67–76.
[12] Cassidy PJ, Bennett HS. TRAMPA multi-depot vehicle scheduling system. J Oper Res Soc 1972;23:151–63.
[13] Chao I, Golden BL, Wasil E. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. Am J Math Manag Sci 1993;13(3–4):371–406.
[14] Gulczynski D, Golden B, Wasil E. The period vehicle routing problem: new heuristics and real-world variants. Transp Res Part E—Log Transp Rev 2011;47:648–68.
[15] Salhi S, Sari M. A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. Eur J Oper Res 1997;103:95–112.

[16] Russell R, Gribbin D. A multi-phase approach to the period routing problem. Networks 1991;21:747–65.
[17] Drummond LA, Ochi LS, Vianna DS. An asynchronous parallel metaheuristic for the period vehicle routing problem. Future Gener Comput Syst 2001;17:379–86.
[18] Renaud J, Laporte G, Boctor FF. A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 1996;23:229–35.
[19] Hemmelmayr VC, Doerner KF, Hartl RF. A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 2009;195:791–802.
[20] Pirkwieser S, Raidl GR. Multilevel variable neighborhood search for periodic routing problems; 2010.
[21] Dondo R, Cerdà J. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. Eur J Oper Res 2007;176:1478–507.
[22] Lau H, Chan T, Tsui W, Pang W. Application of genetic algorithms to solve the multidepot vehicle routing problem. IEEE Trans Autom Sci Eng 2010;7(2):383–92.
[23] Yu B, Yang Z, Xie J. A parallel improved ant colony optimization for multi-depot vehicle routing problem. J Oper Res Soc 2011;62:183–8.
[24] Kang K, Lee Y, Lee B. An exact algorithm for multi-depot and multi-period vehicle scheduling problem; 2005.
[25] Crainic TG, Crisan GC, Gendreau M, Lahrichi N, Rei W. A concurrent evolutionary approach for rich combinatorial optimization. In: Genetic and evolutionary computation conference; 2009. p. 2017–22.
[26] Rahimi-Vahed A, Crainic T, Gendreau M, Rei W. A path relinking algorithm for a multi-depot periodic vehicle routing problem. J Heuristics 2013;19:497–524.
[27] Tan C, Beasley J. A heuristic algorithm for the period vehicle routing problem. Omega—Int J Manag Sci 1984;12:497–504.
[28] Fisher ML, Jaikumar R. A generalized assignment heuristic for the vehicle routing problem. In: Conference on computer networks; 1981.
[29] Christofides N, Beasley JE. The period routing problem. Networks 1984;14(2):237–56.
[30] Toth P, Vigo D. The vehicle routing problem; 2002.
[31] Gendreau M, Hertz A, Laporte G. New insertion and postoptimization procedures for the traveling salesman problem. Oper Res 1992;40:1086–94.
[32] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 2004;31:1985–2002.
[33] Lahrichi N, Crainic T, Gendreau M, Rei W, Rousseau L. Strategic analysis of the dairy transportation problem. In: CIRRELT 80; 2012.
[34] Cordeau J, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 2001;52:928–36.
[35] Coy SP, Golden BL, Runger GC, Wasil EA. Using experimental design to find effective parameter settings for heuristics. J Heuristics 2000;7:77–97.
[36] Smith SK, Eiben AE. Parameter tuning of evolutionary algorithms: generalist vs specialist. 2010.
[37] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 1964;12:568–81.