




A new matheuristic approach for the multi-depot vehicle routing problem with inter-depot routes

Tânia Rodrigues Pereira Ramos^{1,2}  · Maria Isabel Gomes³ · Ana Paula Barbosa-Póvoa¹

Received: 22 February 2018 / Accepted: 5 November 2019 / Published online: 25 November 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

The multi-depot vehicle routing problem with inter-depot routes is studied in this paper, where vehicles may reset their capacity at any depot during the working day. Due to the complexity of this problem, exact approaches are limited to small-size applications. In order to overcome this limitation, we propose a matheuristic which integrates a mixed integer linear programming formulation with a set of relax-and-fix strategies. This solution approach is shown to be very efficient, and for the first time, large-size benchmarking instances are solved.

Keywords Multiple depots · Vehicle routing problem · Inter-depot routes · Mixed integer linear programming · Matheuristic

Electronic supplementary material The online version of this article (<https://doi.org/10.1007/s00291-019-00568-7>) contains supplementary material, which is available to authorized users.

✉ Tânia Rodrigues Pereira Ramos
tania.p.ramos@tecnico.ulisboa.pt

Maria Isabel Gomes
mirg@fct.unl.pt

Ana Paula Barbosa-Póvoa
apovoa@tecnico.ulisboa.pt

- ¹ Centre for Management Studies, Instituto Superior Técnico (CEG-IST), Universidade de Lisboa, Av. Rovisco Pais, 1049-001 Lisbon, Portugal
- ² Business Research Unit, ISCTE (BRU-ISCTE), Instituto Universitário de Lisboa, Lisbon, Portugal
- ³ Centre for Mathematics and Applications, Faculdade de Ciências e Tecnologia (CMA-FCT), Universidade Nova de Lisboa, Lisbon, Portugal

Table 1 Characteristics of five types of routing problems

Depots	Warehouses	Type of problems
Single	None	(0) Vehicle routing problem (VRP)
Single	Single or multiple	(1) Vehicle routing problem with intermediate facilities (VRPIF)
Multiple	None	(2) Multi-depot vehicle routing problem (MDVRP)
Multiple	Single or multiple	(3) Multi-depot vehicle routing problem with intermediate facilities (MDVRPIF)
Multiple	None ^a	(4) Multi-depot vehicle routing problem with inter-depot routes (MDVRPI)

^aDepots act as warehouses for vehicles based at other depots

1 Introduction

Complex logistics systems often entail the use of several depots, where goods are stored and vehicles are allocated, from where they perform distribution activities. Such logistics networks also include storage facilities (like warehouses) that are not directly involved in distribution activities and therefore have no allocated vehicles. The major difference between a depot and a warehouse is that, by convention, the former serves as origin and final destination of vehicles, whereas the latter usually only serves as an intermediate facility within a vehicle's routes. In a multi-facility scenario, this distinction leads to four different routing problems (see Table 1 and Fig. 1):

- (1) the vehicle routing problem with intermediate facilities (VRPIFs) where one facility acts as depot and the remainder as warehouses,
- (2) the multi-depot vehicle routing problem (MDVRP) where all facilities act only as depots,
- (3) the multi-depot vehicle routing problem with intermediate facilities (MDVRPIFs) where some facilities act as depots and some as warehouses,
- (4) the multi-depot vehicle routing problem with inter-depot routes (MDVRPIs) where all facilities can act simultaneously as depots and as warehouses.

A VRP (see Table 1) is characterized by a logistics network with a single depot and no warehouses. The VRP has been extensively studied in the literature, and several algorithms have been proposed to solve it. Laporte (2009) and Toth and Vigo (2014) give a comprehensive overview of the problem. Due to its combinatorial nature, several heuristics have been developed to solve this problem, spanning from classical heuristics to metaheuristics and, more recently, matheuristics. Classical heuristics were classified by Laporte and Semet (2002) into three categories: constructive [e.g. Clarke and Wright (1964)]; two-phase [e.g. Gillet and Miller (1974), Renaud et al. (1996a)]; and improvement heuristics (e.g. Lin (1965), Lin and Kernighan (1973), Van Breedam (2001)]. Metaheuristics were classified by Laporte (2009) into three categories: local search (e.g. tabu search, simulated annealing and variable neighbourhood search); evolutionary algorithms (e.g. genetic algorithms

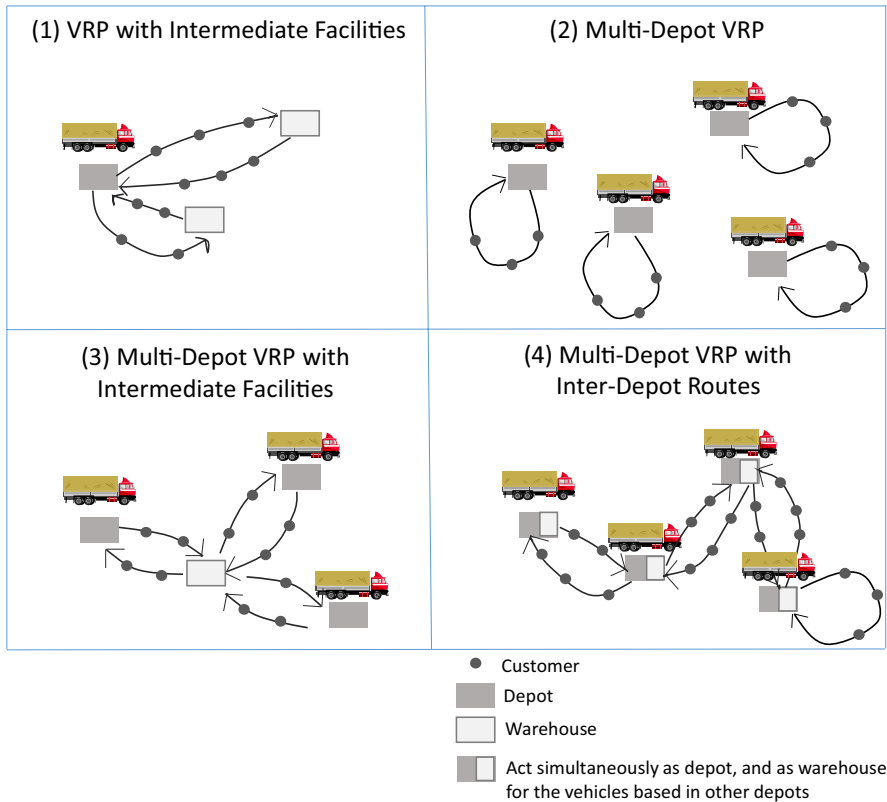


Fig. 1 Schematic representation of four types of routing problems

and adaptive memory); and learning mechanisms (e.g. neural networks and ant colony optimization). Matheuristics are the result of a hybridization between heuristics and mathematical programming (Maniezzo et al. 2009), which exploit mathematical programming techniques within a heuristic framework.

Additionally, for the single-depot case (see Table 1 and Fig. 1), the existence of intermediate facilities allows drivers to reset the capacity of their vehicles during a shift without having to return to the central depot. In such logistics systems, a feasible route sequence starts at a central depot, visits a subset of customers, may refill at one of the intermediate facilities before visiting more customers, and returns to the depot. Refilling is allowed when required and vehicles only return to the original depot at the end of the route. The refill may occur at an intermediate facility or at the central depot. The same vehicle can perform multiple trips per day, which relates this problem with the Multi-Trip VRP. Several methods have been proposed in the literature for solving the Multi-Trip VRP. Brandao and Mercer (1997) developed a tabu search, Petch and Salhi (2003) developed a constructive heuristic, Mingozzi et al. (2013) developed an exact algorithm based on the set partitioning formulation,

and Cattaruzza et al. (2014) proposed a memetic algorithm, just to name a few works.

The term “intermediate facilities” was first introduced by Angelelli and Speranza (2002); however, some years prior, Bard et al. (1998) used the term “satellite facilities”. Note that, “intermediate facilities” and “satellite facilities” are considered as having the same meaning in logistics. Bard et al. (1998) introduced the VRP with satellite facilities as a mixed integer linear programming formulation and developed a branch-and-cut algorithm to solve it. Angelelli and Speranza (2002) tackled the same problem, calling them intermediate facilities instead of satellite facilities, in a collection context. In this problem, a vehicle departs from the depot to collect goods from customers. When the vehicle is at full capacity, it heads towards an intermediate facility in order to unload the goods, after which it may start another collection route until the work shift ends, when it must return to the central depot. The authors develop a tabu search algorithm to solve the periodic VRP with intermediate facilities and present computational results on a set of instances. Similar collection problems have also been addressed by Kim et al. (2006), Benjamin and Beasley (2010), and Hemmelmayr et al. (2013). Tarantilis et al. (2008) also addressed the VRP with intermediate facilities and proposed a three-step algorithmic framework to solve the problem, which is based on a set of metaheuristic procedures. The developed approach was applied to a set of large-scale test instances.

Finally, the existence of multiple depots allows vehicles to be based at different depots (instead of at a central depot). In the classical MDVRP, routes start and end at the same depot, and only customers are visited in between. Stops at intermediate facilities are not allowed. Several works have been published addressing this problem, where mainly heuristic approaches have been developed [see Chao et al. (1993), Renaud et al. (1996b), Cordeau et al. (1997), Lim and Wang (2005), Dondo and Cerda (2009), Vidal et al. (2012), Tu et al. (2014)]. Using exact approaches, the works of Laporte et al. (1984, 1988) and Baldacci and Mingozzi (2009) also solved the MDVRP problem. The introduction of intermediate facilities can be considered an extension of the classical MDVRP. This problem has been modelled either as a MDVRP with intermediate facilities (where only warehouses act as intermediate points), or as a MDVRP with inter-depot routes (when depots, other than the vehicle’s home depot, also act as intermediate points). To the best of the authors’ knowledge, the MDVRP with intermediate facilities has only been addressed in the work of Markov et al. (2016), while the MDVRP with inter-depot routes was introduced by Crevier et al. (2007) and later studied by Muter et al. (2014).

Markov et al. (2016) addressed a real case of a complex recyclable waste collection system. In this problem, each vehicle route starts and ends at one of several depots, not necessarily the same for each vehicle. A sequence of collections followed by disposals at the available dumps (the intermediate facilities) was observed. In this case, since several depots are involved, the problem can be framed as a MDVRP instead of VRP. In addition, given that all routes must pass through a dump (intermediate facility) before returning to a depot, the problem is classified as a MDVRP with intermediate facilities. With the introduction of a flexible assignment of destination depots, it follows that a vehicle can start and end its route at different depots. A mathematical model based on the three-index formulation was developed, and

as solution approach, a multiple neighbourhood search heuristic was designed and tested on literature instances, leading to the solution of a real case.

Crevier et al. (2007) introduced the MDVRP with inter-depot routes and the concept of “rotation”. A rotation is defined as “the set of routes assigned to the same vehicle, which are constrained by a preset maximum duration”. A rotation has to start and end at the same depot and may include single-depot routes (a route starting and ending at the same depot), inter-depot routes (a route connecting two different depots) or a combination of the two. Crevier et al. (2007) proposed a set partitioning approach to solve the problem. This requires an a priori definition of all single-depot and inter-depot routes, a so-called herculean task. The solution method developed by the authors entailed the use of a tabu search heuristic, previously proposed by Cordeau et al. (1997), in order to select the set of routes to be considered in the partitioning formulation. Given the novelty of the work, new benchmark instances were generated to demonstrate the model’s adequacy. Under the assumption that it is rarely cost-effective to use inter-depot routes when vehicles are based at different depots, the authors considered, in the solved instances, that vehicles were solely located at the central depot and that all other depots were used only as intermediate replenishment facilities. In fact, Crevier et al. (2007) solved a problem that was later re-named as VRPIF by Tarantilis et al. (2008), to emphasize “both the replenishment role of the intermediate facilities and the use of a single central station for the fleet of vehicles”. Later, Muter et al. (2014) tackled the MDVRPI where vehicles could be based at multiple depots rather than at a single central depot. A branch-and-price algorithm with two different pricing strategies was proposed and tested on new instances based on the benchmarking instances developed by Crevier et al. (2007). These instances have 25 and 40 customers, and 4 and 6 vehicles, respectively. Summing up, Crevier et al. (2007) developed benchmarking instances for the MDVRPI but solved them as a VRPIF, while Muter et al. (2014) solved the MDVRPI but only for smaller instances (with 25 and 40 customers). Thus, the original benchmarking instances developed for the MDVRPI (with 48–288 customers) have not yet been solved as a MDVRPI, in which vehicles are based in multiple depots. This establishes a research opportunity.

In this paper, we aim to explore this research opportunity and our contribution is threefold: Firstly, we propose a new formulation for the MDVRPI through a flow formulation, where both routes and rotations are simultaneously defined by the model. Therefore, we overcome the need of generating routes beforehand. We define “rotation”, “single-depot route” and “inter-depot route” as defined by Crevier et al. (2007): “the set of all routes assigned to a vehicle is called a rotation whose total duration cannot exceed a preset value”, “a single-depot route starts and ends at the same depot while an inter-depot route connects two different depots”. Our model addresses tactical-operational decisions as the strategic decisions regarding the number and location of the facilities in the network were already defined, but the role of each facility (depot, intermediate facility or both), as well as the home depot for each vehicle, is to be defined while simultaneously defining the routing plan to be implemented for a medium-term period.

Secondly, to overcome the computational burden associated with the type of problem being studied, we develop a matheuristic procedure that explores a

decomposition approach based on the relaxation of some constraints of the full proposed mathematical model. This matheuristic is capable of solving the majority of the instances used by Muter et al. (2014), in addition to the original MDVRPI instances with vehicles based at multiple depots which, to the best of our knowledge, were solved for the first time in this work. We choose this specific type of heuristic because we want to explore the mathematical formulation as much as possible, since this provides a generic form of structuring and formulating these problems, exploring the problem characteristics. The advantages of matheuristics are, on the one hand, granting to mathematical programming approaches the robustness and time effectiveness that characterize heuristics and, on the other hand, exploiting the mathematical programming model formulation in the customization of a heuristic for specific problems.

Lastly, and considering the applicability of the problem in hand, the economic benefit of stationing vehicles at multiple depots rather than at a central depot is assessed. The characteristics of a MDVRPI occur in real problems such as grocery distribution problems as addressed by Crevier et al. (2007); recyclable waste collection networks with multiple facilities, where trucks may unload the collected recyclable waste at any depot of the network; or the management of an electric fleet of vehicles, which have to recharge their batteries during a working day (Schneider et al. 2015).

The remainder of the paper is structured as follows: in Sect. 2, the mathematical formulation for the MDVRPI is presented, followed by the discussion of the solution methodology developed in Sect. 3. Computational results obtained for benchmark instances are reported in Sect. 4. Finally, Sect. 5 concludes the paper and proposes some suggestions for further research.

2 Problem formulation

The MDVRPI can be defined as the problem of designing a set of vehicle rotations which optimize a predetermined objective, such as the minimization of a total routing cost, while assuring that: (1) a rotation starts and ends at the same depot; (2) the duration of a rotation (including travel, service and loading times) does not exceed a preset limit; (3) a route starts at a certain facility but may end in a different one; (4) each customer is visited exactly once; and (5) the total demand covered by each route does not exceed the vehicle capacity. Note that, while a rotation must start and end at the depot where vehicles are parked, routes can start and/or end at any type of facility. Any facility may act as depot, as an intermediate facility or as both.

The MDVRPI problem is formulated in this work exploring the two-commodity flow formulation concepts (Baldacci et al. 2004). An undirected graph $G = (V, E)$ is considered, where $V = V_c \cup V_f \cup V_g$, being $V_c = \{1, \dots, n\}$ a set of n customers, $V_f = \{n + 1, \dots, n + w\}$ a set of w facilities, $V_g = \{n + w + 1, \dots, n + 2w\}$ a replica of the facility set and $E = \{(i, j) : i, j \in V_c \cup V_f \cup V_g, i \neq j\}$ the edge set. The facility replica set is needed because, in the two-commodity flow formulation, routes are defined by paths between the real facilities and the replica ones. To establish the

routes, this formulation requires two flow variables that, in turn, define two flow paths for any route. One of the paths, from the real facilities to the replica ones, designed through the flow variable, represents the load of the vehicle, which decreases along the route in a distribution problem. The other path, from the replica facilities to the real ones, corresponds to the second flow variable and represents the empty space on the vehicle, which increases along the route.

Each customer is characterized by their demand P_i and service time S_i . Each route k of set $K = \{1, \dots, s\}$ has a capacity of Q units, and each rotation r of set $R = \{1, \dots, m\}$ has a maximum duration of T . Each edge (i, j) has an associated cost C_{ij} and a travel time F_{ij} . We also consider a loading time L to account for the time it takes to fully load a vehicle at the start of each route.

The following decision variables are defined:

$$x_{ijk r} \begin{cases} 1, & \text{if edge } (i, j) \text{ is traversed on route } k \text{ that belongs to rotation } r \\ 0, & \text{otherwise} \end{cases}$$

$y_{ijk r}$ Flow variable representing the vehicle load when travelling from nodes i to j on route k that belongs to rotation r . The flow $y_{ijk r}$ represents the empty space on vehicle route k that belongs to rotation r

$$z_{ik r} \begin{cases} 1, & \text{if node } i \text{ is visited by route } k \text{ on rotation } r \\ 0, & \text{otherwise} \end{cases}$$

$e_{ijk r}$ Exit time from node i to node j on route k on rotation r

$a_{ijk r}$ Arrival time at node j from node i on route k on rotation r

$$g_{ik r} \begin{cases} 2, & \text{if route } k \text{ starts and ends at depot on rotation } r \\ 1, & \text{if route } k \text{ starts or ends at depot on rotation } r \\ 0, & \text{otherwise} \end{cases}$$

The mathematical formulation also includes the following auxiliary variables:

$$\alpha_{ik r} \begin{cases} 1, & \text{if route } k \text{ does not start nor ends at depot on rotation } r \\ 0, & \text{otherwise} \end{cases}$$

$$\beta_{ik r} \begin{cases} 1, & \text{if route } k \text{ starts or ends at depot on rotation } r \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ik r} \begin{cases} 1, & \text{if route } k \text{ starts and ends at depot on rotation } r \\ 0, & \text{otherwise} \end{cases}$$

$$\gamma_{ir} \begin{cases} 1, & \text{if rotation } r \text{ starts at depot } i \\ 0, & \text{otherwise} \end{cases}$$

The solution for the MDVRPI is a set of at most m constrained cycles (called rotations) on graph G which minimizes the total cost while ensuring that every customer is visited. The cycles are smaller than or equal to a maximum duration T , and each segment of the cycle (single-route or inter-depot route) is feasible with respect to the vehicle capacity Q . Figure 2 depicts a feasible solution for the MDVRPI, where three rotations are defined through eight routes. Rotation 1 includes three routes (one single-depot and two inter-depot routes—in blue), rotation 2 has two routes (all single-depot routes—in black) and rotation 3 has three routes (all inter-depot routes—in orange).

The MDVRPI formulation is then:

$$\text{minimize } \frac{1}{2} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} \sum_{r \in R} x_{ijkr} C_{ij} \tag{1}$$

$$\text{subject to } \sum_{j \in V} (y_{jikr} - y_{ijkr}) = 2P_i z_{ikr}, \quad \forall i \in V_c, k \in K, r \in R \tag{2}$$

$$\sum_{i \in V_j} \sum_{j \in V_c} \sum_{k \in K} \sum_{r \in R} y_{ijkr} = \sum_{j \in V_c} P_j \tag{3}$$

$$\sum_{i \in V_j} \sum_{j \in V_c} \sum_{k \in K} \sum_{r \in R} y_{ijkr} \leq |K|Q - \sum_{j \in V_c} P_j \tag{4}$$

$$\sum_{j \in V_c} y_{ijkr} \leq Q, \quad \forall i \in V_g, k \in K, r \in R \tag{5}$$

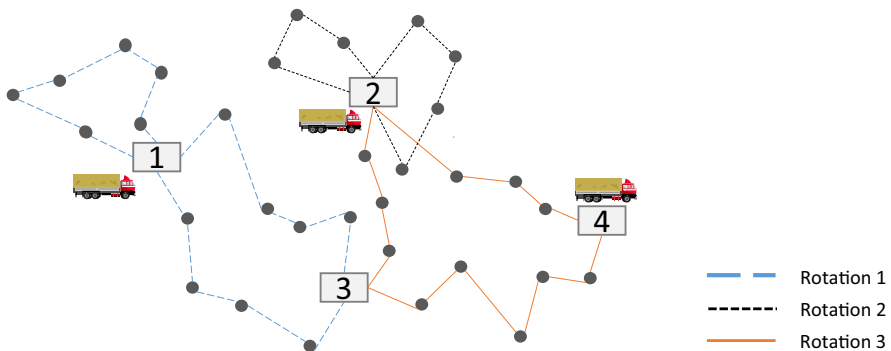


Fig. 2 Representation of a feasible solution for the MDVRPI (color figure online)

$$y_{ijk_r} + y_{jik_r} = Qx_{ijk_r}, \quad \forall i, j \in V, k \in K, r \in R \tag{6}$$

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ijk_r} = 2z_{jkr}, \quad \forall j \in V_c, k \in K, r \in R \tag{7}$$

$$\sum_{k \in K} \sum_{r \in R} z_{ikr} = 1, \quad \forall i \in V_c \tag{8}$$

$$\sum_{r \in R} z_{ikr} \leq 1, \quad \forall i \in V_c, k \in K \tag{9}$$

$$\sum_{k \in K} z_{ikr} \leq 1, \quad \forall i \in V_c, r \in R \tag{10}$$

$$\sum_{k \in K} \sum_{r \in R} x_{ijk_r} \leq 1, \quad \forall i, j \in V, i \neq j \tag{11}$$

$$\sum_{i \in V_f} \sum_{j \in V} x_{ijk_r} \leq 1, \quad \forall k \in K, r \in R \tag{12}$$

$$e_{ijk_r} + F_{ij}x_{ijk_r} = a_{ijk_r}, \quad \forall i, j \in V, i \neq j, \forall k \in K, r \in R \tag{13}$$

$$\sum_{\substack{i \in V \\ i \neq j}} (e_{jik_r} - a_{ijk_r}) = 2S_j z_{jkr}, \quad \forall j \in V_c, k \in K, r \in R \tag{14}$$

$$e_{ijk_r} - \sum_{h \in V \setminus \{i, j\}} a_{hik_r} \leq S_i x_{ijk_r}, \quad \forall i \in V_c, \forall j \in V, i \neq j, \forall k \in K, r \in R \tag{15}$$

$$e_{ijk_r} \leq Tx_{ijk_r}, \quad \forall i, j \in V, k \in K, r \in R \tag{16}$$

$$a_{ijk_r} \leq Tx_{ijk_r}, \quad \forall i, j \in V, k \in K, r \in R \tag{17}$$

$$\sum_{j \in V} \sum_{i \in V_g} \sum_{k \in K} a_{jik_r} + \sum_{j \in V} \sum_{i \in V_g} \sum_{k \in K} x_{jik_r} L \leq T, \quad \forall r \in R \tag{18}$$

$$\sum_{j \in V} \sum_{k \in K} \sum_{r \in R} x_{ijk_r} + x_{jik_r} = \sum_{j \in V} \sum_{k \in K} \sum_{r \in R} x_{(i+w)jkr} + x_{j(i+w)kr}, \quad \forall i \in V_f \tag{19}$$

$$g_{ikr} = \sum_{\substack{j \in V \\ j \neq i}} x_{ijk} + \sum_{\substack{j \in V \\ j \neq i}} x_{j(i+w)kr}, \quad \forall i \in V_f, k \in K, r \in R \quad (20)$$

$$g_{ikr} = \beta_{ikr} + 2\delta_{ikr}, \quad \forall i \in V_f, k \in K, r \in R \quad (21)$$

$$\alpha_{ikr} + \beta_{ikr} + \delta_{ikr} = 1, \quad \forall i \in V_f, k \in K, r \in R \quad (22)$$

$$g_{ikr} \leq \sum_{\substack{k' \in K \\ k' \neq k}} \beta_{k'ir} + 2\gamma_{ir}, \quad \forall i \in V_f, k \in K, r \in R \quad (23)$$

$$\sum_{i \in V_f} \gamma_{ir} \leq 1, \quad \forall r \in R \quad (24)$$

$$y_{ijk} \geq 0, \quad \forall i, j \in V, k \in K, r \in R \quad (25)$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in V, k \in K, r \in R \quad (26)$$

$$z_{ikr} \in \{0, 1\}, \quad \forall i \in V_c, k \in K, r \in R \quad (27)$$

$$e_{ijk}, a_{ijk} \geq 0, \quad \forall i, j \in V \quad (28)$$

$$g_{ikr} \in \{0, 1, 2\}, \quad \forall i \in V_f, k \in K, r \in R \quad (29)$$

$$\alpha_{ikr}, \beta_{ikr}, \delta_{ikr} \in \{0, 1\}, \quad \forall i \in V_f, k \in K, r \in R \quad (30)$$

$$\gamma_{ir} \in \{0, 1\}, \quad \forall i \in V_f, r \in R \quad (31)$$

The objective function (1) minimizes the total routing cost where, due to the two flow paths that define a route, each edge is counted twice. Therefore, the total routing cost has to be divided by two to translate the actual cost.

Constraints (2)–(6) model the flows that implicitly define the routes. Constraint (2) models the inflows and outflows for each customer, assuring that the inflow minus the outflow equals twice the demand, since two paths cross each customer. The total outflow of real facilities must equal the total demand [Constraint (3)], and the total inflow is less than or equal to the residual capacity of the used routes [Constraint (4)]. The cardinality of route set K must be sufficiently large not to constrain the solution. Route capacity must not be exceeded, and this is guaranteed by Constraints (5) and (6). Constraint (7) assures that any feasible solution contains two

incident edges for each customer node due to the two paths that characterize each route. Constraint (8) ensures that all customers are visited.

Constraints (9)–(11) relate routes with rotations, ensuring that all routes belong to a rotation. Constraint (12) guarantees that each route starts, at most, at one of the real facilities.

Constraints (13)–(18) model the duration of routes and rotations. The arrival time at each node is equal to the exit time from the previous node plus the travel time F_{ij} [Constraint (13)]. The difference between the exit time and the arrival time within each customer is equal to the service time [Constraint (14)]. Time continuity is ensured by Constraint (15). If edge (i, j) is not crossed, then the arrival and exit times of that edge are equal to zero [Constraints (16) and (17)]. Constraint (18) guarantees that the maximum time for a rotation is not exceeded.

Since inter-depot routes may be a part of the solution for this problem, Constraint (19) ensures route continuity among inter-depot routes by enabling a rotation in these cases. Therefore, the number of outbound edges at each real facility must equal the number of inbound edges at the corresponding replica facility.

Constraints (20)–(22) define the values for variables g_{ikr} , α_{ikr} , β_{ikr} and δ_{ikr} . Variable g_{ikr} assumes the value two if route k starts at facility i and ends at the same facility replica. If route k only starts or ends at facility i , g_{ikr} takes value one [Constraint (20)]. Variable g_{ikr} is also a function of binary variables α_{ikr} , β_{ikr} and δ_{ikr} [Constraint (21)], which act as discretization variables. Notice that variable α_{ikr} can be omitted since it models g_{ikr} when it is equal to zero. Being discretization variables, only one of them can assume the value one [Constraint (22)]. Constraint (23) ensures that if a closed route belongs to the solution (whenever $g_{ikr} = 2$), either two inter-depot routes are part of the same rotation or the rotation starts at depot i . Each rotation must only start at one depot [Constraint (24)]. Finally, Constraints (25)–(31) define the variables' domains.

3 Solution Methodology

To solve the MDVRPI, we develop a matheuristic approach, which adopts a relax-and-fix strategy supported by the previously presented formulation (see Fig. 3). First, the duration constraints and the number of vehicles available are relaxed, and then, if the solution satisfies the relaxed constraints, the corresponding variables are fixed. The main idea behind this relaxation step is to solve a less constrained problem, which is expectedly easier to solve. Moreover, the relaxed problem also provides a lower bound for the MDVRPI.

The matheuristic is composed of four main modules, where a different mathematical formulation is solved in each one, plus a post-optimization module:

- (1) Module 1 solves a MDVRPI Relaxation I (MDVRPI without duration constraints and unlimited vehicle fleet). This module provides the lower bound;
- (2) Module 2 solves an Assignment Problem, where routes are assigned to rotations;
- (3) Module 3 solves a MDVRPI Relaxation II (MDVRPI with duration constraints and unlimited vehicle fleet);

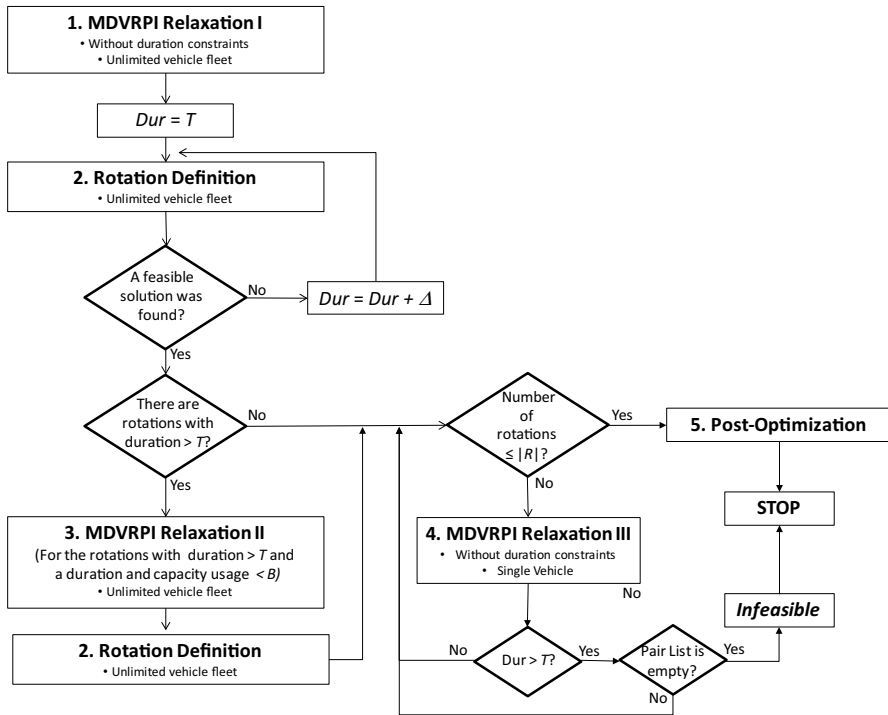


Fig. 3 Matheuristic solution methodology flowchart

- (4) Module 4 solves a MDVRPI Relaxation III (MDVRPI without duration constraints and single vehicle/rotation);
- (5) Module 5 applies 2-opt and 3-opt moves within and between rotations.

Analysing the matheuristic shown in Fig. 3, and starting with module 1, this takes the concept of rotation into account while relaxing the time limit for the vehicle to return to its original depot. The number of available vehicles is also relaxed (equivalent to the number of rotations). Single- and inter-depot routes are then designed without duration concerns, while guaranteeing that inter-depot routes are linked in order to enable rotations. Afterwards, the duration of each route is assessed, allowing the definition of vehicle rotations to take place in the next module. The maximum duration for a rotation (parameter Dur) is initially set to the value T . Module 2 is then run to define rotations. If there is a feasible solution with $Dur = T$, the procedure continues, and if not, then the maximum duration (Dur) is incremented by a value Δ until a feasible solution is obtained. When such a solution is reached (with a duration for a rotation exceeding the predefined limit of T), module 3 is executed, imposing the duration constraints that were relaxed in module 1. This third module is executed considering both sites that belong to rotations exceeding the time limit, and the sites belonging to rotations have a low duration and capacity usage.

After module 3, module 2 is executed once again, in order to re-define rotations considering all routes that were produced until then and the maximum time limit allowed (T). If the number of rotations obtained is smaller than or equal to the number of available rotations (i.e. the number of available vehicles), a feasible solution is reached, and therefore, the solution procedure moves to the final module, where some post-optimization moves are applied to each rotation and between rotations. Otherwise, module 4 merges two rotations into a single rotation until the number of rotations equals the number of available vehicles. All pairs of rotations that can be merged are listed (called Pair List), and after a pair is merged into one rotation, the duration is assessed. If the duration is higher than allowed (T), then the merge process is considered for the next pair of rotations. If all pairs have been submitted to module 4 (Pair List is empty) without complying with the duration limit, that means the matheuristic was not able to produce a feasible solution.

The mathematical formulations and details for each one of the five modules are provided below and in supplementary material: Appendix A.

3.1 MDVRPI Relaxation I

The relaxed version I of the MDVRPI can be formulated as follows: Let $G = (V_c \cup V_f \cup V_g, E)$ be an undirected graph, where $V_c = \{1, \dots, n\}$ is the set of customers, $V_f = \{n + 1, \dots, n + w\}$ is the set of facilities, $V_g = \{n + w + 1, \dots, n + 2w\}$ is the set of facility replicas and $E = \{(i, j) : i, j \in V_c \cup V_f \cup V_g, i \neq j\}$ is the set of edges. A demand P_i is associated with customer i while a travel cost C_{ij} with the edge (i, j) . An unlimited fleet of vehicles with capacity Q is available, and each vehicle route has a fixed cost H . This formulation uses binary variables x_{ij} equal to 1 if edge (i, j) is traversed, a flow variable y_{ij} representing the vehicle load when traveling from node i to node j and an integer decision variable k representing the number of vehicle routes in the solution. The feasible solution is a set of routes (single-depot routes and/or inter-depot routes) which minimize the total cost (travel cost plus a fixed cost for each vehicle route created), ensuring that every customer is visited and the capacity of the vehicles is not surpassed. The formulation is given below, whereas Constraints (33)–(39) correspond to Constraints (2)–(7) and (19) of the full model.

$$\text{minimize } \frac{1}{2} \sum_{i \in V} \sum_{j \in V} x_{ij} C_{ij} + H.k \tag{32}$$

$$\text{subject to } \sum_{\substack{j \in V \\ j \neq i}} (y_{ji} - y_{ij}) = 2P_i, \quad \forall i \in V_c \tag{33}$$

$$\sum_{i \in V_f} \sum_{j \in V_c} y_{ij} = \sum_{j \in V_c} P_j \tag{34}$$

$$\sum_{i \in V_f} \sum_{j \in V_c} y_{ji} = k.Q - \sum_{j \in V_c} P_j \quad (35)$$

$$\sum_{i \in V_g} \sum_{j \in V_c} y_{ij} = k.Q \quad (36)$$

$$y_{ij} + y_{ji} = Qx_{ij}, \quad \forall i, j \in V, i \neq j \quad (37)$$

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 2, \quad \forall j \in V_c \quad (38)$$

$$\sum_{j \in V_c} x_{ij} + \sum_{j \in V_c} x_{ji} = \sum_{j \in V_c} x_{(i+w)j} + \sum_{j \in V_c} x_{j(i+w)}, \quad \forall i \in V_f \quad (39)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \quad (40)$$

$$y_{ij}, y_{ji} \geq 0, \quad \forall i, j \in V \quad (41)$$

$$k \text{ integer} \quad (42)$$

Since this formulation disregards the route durations, at a post-processing phase, the duration of each route is assessed.

3.2 Rotation Definition

Let K denote the set of all routes defined in the previous module ($K = \{1, \dots, s\}$), V_f the set of depots ($V_f = \{1, \dots, w\}$) and R the set of rotations ($R = \{1, \dots, m\}$). The goal of this second module is to assign each route k to a rotation r . SD_{ik} and O_{ik} define a partition of set K , where $SD_{ik} \subseteq K$ is the set of routes starting and ending at depot i (single-depot routes) and $O_{ik} \subseteq K$ is the set of routes with one depot in i and the other depot outside i (inter-depot routes). Each route k is characterized by: (1) a duration D_k (including travel, service and loading times) and (2) the definition of the start and end depot given by parameter G_{ik} . This parameter acts as the variable g_{ikr} of the full model. Therefore, when $G_{ik} = 2$, route k starts and ends at depot i (single-depot route); when $G_{ik} = 1$, route k starts or ends at depot i (inter-depot route) and when $G_{ik} = 0$, route k neither starts nor ends at depot i . The maximum duration allowed for a rotation is given by parameter Dur .

As mentioned, this formulation assigns routes to rotations, where ε_{kr} is a binary variable equal to 1 if route k is assigned to rotation r . If rotation r is in the solution, the binary variable π_r equals 1. Otherwise, $\pi_r = 0$. Two auxiliary variables need to be defined: an integer variable μ_{ir} that models the number of times rotation r visits

depot i in an inter-depot route and the binary variable γ_{ir} which is equal to 1 if rotation r starts at depot i . Notice that the cardinality of set R is sufficiently large not to constrain the solution, since an unlimited vehicle fleet is considered in this module. The solution is the minimum set of rotations which comply with the maximum duration allowed. The mathematical formulation to define rotations is as follows:

$$\text{minimize } \sum_{r \in R} \pi_r \tag{43}$$

$$\text{subject to } \sum_{k \in O_{ik}} G_{ik} \varepsilon_{kr} - 2\mu_{ir} = 0, \quad \forall r \in R, \forall i \in V_f \tag{44}$$

$$\sum_{k \in C_{ki}} \varepsilon_{kr} \leq |SD_{ik}| \left(\sum_{k \in O_{ki}} \varepsilon_{kr} + \gamma_{ir} \right), \quad \forall i \in V_f, \forall r \in R \tag{45}$$

$$\sum_{k \in K} D_k \varepsilon_{kr} \leq \text{Dur}, \quad \forall r \in R \tag{46}$$

$$\sum_{r \in R} \varepsilon_{kr} = 1, \quad \forall k \in K \tag{47}$$

$$\varepsilon_{kr} \leq \pi_r, \quad \forall k \in K, \forall r \in R \tag{48}$$

$$\sum_{i \in V_f} \gamma_{ir} \leq \pi_r, \quad \forall r \in R \tag{49}$$

$$\varepsilon_{kr}, \pi_r, \gamma_{ir} \in \{0, 1\} \quad \forall k \in K, \forall r \in R, \forall i \in V_f \tag{50}$$

$$\mu_{ir} \text{ integer } \quad \forall r \in R, \forall i \in V_f \tag{51}$$

The objective function (43) minimizes the number of rotations, given that the number of rotations available is unlimited. Constraints (44)–(46) are based on the work of Crevier et al. (2007). Constraint (44) guarantees rotation continuity when inter-depot routes are in a rotation, while Constraint (45) ensures either rotation continuity or rotation starting when single-depot routes are in a rotation. Constraint (46) limits the total duration of a rotation. Constraint (47) guarantees that all routes are assigned to a rotation. Constraints (48) and (49) state that a rotation is used if a route is assigned to it or if a rotation starts at a depot.

The second module is solved through an iterative procedure concerning the total duration of a rotation (parameter Dur). The goal of this procedure is to have the maximum number of rotations which satisfy the maximum duration (T) in order to minimize the number of rotations to be worked out by module 3. Recall that, in the first module, no duration constraints were imposed, so solutions where the maximum

duration for a rotation is exceeded might be generated. To deal with this situation, parameter *Dur* is initialized as *T*, and if there is no feasible solution within this time limit, then the value of *Dur* is iteratively increased by Δ until a feasible solution is obtained. The value for Δ should be the smallest possible in order to obtain rotations with a duration close to *T*. Since rotations are combinations of routes, we suggest the value Δ to be the minimum duration of the routes ($\Delta = \min\{D_k : k \in K\}$), to allow the inclusion of, at least, one route in a rotation and, therefore, to reach a feasible solution.

3.3 MDVRPI Relaxation II

This module is activated if and only if rotations with a duration larger than *T* are produced in the previous module. The relaxed version II of the MDVRPI can be formulated as follows: Let $G = (V'_c \cup V_f \cup V_g, E)$ be an undirected graph, where $V'_c \subseteq V_c$ is a subset of customers that belong to a rotation with *Dur* > *T*, V_f is the set of facilities, V_g is the set of facility replicas and $E = \{(i, j) : i, j \in V'_c \cup V_f \cup V_g, i \neq j\}$ is the edge set. Let *K* denote the set of routes to be created by this module. A demand P_i and a service time S_i are assigned to customer $i \in V'_c$, and a travel cost C_{ij} and a travel time F_{ij} are associated with the edge (i, j) . An unlimited fleet of vehicles with capacity *Q* is available. Let *L* be the fixed duration representing the time needed for a vehicle to dock at a depot and *T* be the maximum duration for a rotation. This formulation uses binary variables x_{ijk} equal to 1 if edge (i, j) is traversed by route *k*, z_{ik} equal to 1 if customer *i* is visited by route *k*, a flow variable y_{ijk} representing the vehicle load when travelling from nodes *i* to *j* on route *k*, and two positive variables e_{ijk} and a_{ijk} representing the exit time and the arrival time, respectively, from node *i* to node *j* on route *k*. The solution is a set of routes (single-depot routes and/or inter-depot route) which minimize the total cost (travel cost), ensuring that every customer is visited, the capacity of the vehicles is not surpassed and the duration of either single-depot routes and linked inter-depot routes (enabling a rotation) satisfies the maximum duration allowed (*T*).

This formulation is composed of Constraints (1)–(8) and (11)–(19) considering the decision variables mentioned (i.e. without index *r*). In order to guarantee that the sum of the duration of the linked inter-depot routes does not exceed the maximum duration for a rotation, Constraint (52) is added.

$$\sum_{i \in V} \sum_{j, j' \in \xi} \sum_{k \in U} (a_{ijk} - \text{BigM}(1 - z_{(j'-w)k}) + a_{i(j'-w)k} - \text{BigM}(1 - z_{jk})) \leq 2(T - |U|L),$$

$$\xi \subseteq V_g, 2 \leq |\xi| \leq w, U \subseteq K, 2 \leq |U| \leq s$$
(52)

Constraint (52) is inspired by how the Dantzig–Fulkerson–Johnson constraints work to eliminate subtours (Dantzig et al. 1954). This type of constraints has the drawback of leading to an exponential number of additional constraints. However, it is often the case that not all inequalities need to be added to the formulation at the beginning. They can be generated only as required by a separation algorithm, meaning that the formulation solution can start without Constraint (52), and then,

if a rotation with a duration larger than T occurs, a new cut is added to avoid that rotation.

At first, only the demand sites that belong to rotations exceeding T were considered. However, in preliminary tests, it was noted that better solutions were attained if the customers belonging to rotations with a low usage rate were also considered in this module. This is due to better rotations being defined by combining those which are overloaded with those which are under loaded. Therefore, the demand sites that belong to rotations with a usage rate smaller than B are also added. The usage rate accounts for duration and capacity simultaneously, i.e. only the rotations with both usage rates below B are included in this module. The value of B varies in the interval $[0, 1]$. A value near 1 will include all rotations in module three, which will have a negative impact on the computational results given the combinatorial complexity of the problem. A value near 0 implies no additional rotations for module three, which may result in a negative impact on the quality of the solution.

3.4 MDVRPI Relaxation III

As stated before, the previous modules have considered an unlimited vehicle fleet, which represents a relaxation of the original problem. Thus, if the number of rotations produced by the matheuristic exceeds the number of vehicles available, module four must be run in order to decrease that number. This is done by merging rotations: firstly, all pairs of rotations that could be merged are assessed according to their combined duration, and then, the feasible pairs are ordered regarding the travel distance (considering the customers and the depots involved), from the nearest rotations to the farthest rotations. In case of a similar distance, the second ordering criterion is the combined duration, from the lower duration rotations to the higher ones. The first pair of rotations on the list is chosen to be merged through the relaxed version III of the MDVRPI.

The relaxed version III of the MDVRPI can be formulated as follows: Let $G = (V'_c \cup V'_f \cup V'_g, E)$ be an undirected graph, where $V'_c \subseteq V_c$ is a subset of customers that belong to the two rotations to be merged, $V'_f \subseteq V_f$ is the subset of facilities involved in the two rotations, $V'_g \subseteq V_g$ is the subset of facility replicas, and $E = \{(i, j) : i, j \in V'_c \cup V'_f \cup V'_g, i \neq j\}$ is the edge set. Let K denote the set of routes to be created at this module. Each route has a capacity Q . A demand P_i is assigned to customer $i \in V'_c$, and a travel cost C_{ij} is associated with the edge (i, j) . This formulation uses binary variables x_{ijk} and z_{ik} , integer variables g_{ik} equal to 2 if route k starts and ends at depot i , equal to 1 if route k starts or ends at depot i and equal to 0 otherwise. Three auxiliary binary variables are also introduced: δ_{ik} , β_{ik} , α_{ik} . δ_{ik} is equal to 1 if route k starts and ends at depot i , β_{ik} is equal to 1 if route k starts or ends at depot i and α_{ik} is equal to 1 if route k does not start or end at depot i .

The solution is a set of routes (single-depot routes and/or inter-depot route) which minimize the total cost (travel cost), ensuring that every customer is visited, the capacity of the vehicles is not surpassed and all defined routes form a cycle enabling one single rotation. This formulation is composed by Constraints (1)–(8), (12),

and (19)–(22) considering the decision variables without index r . Constraint (23) is rewritten as follows:

$$g_{ik} \leq \sum_{\substack{k' \in K \\ k' \neq k}} \beta_{ik'}, \quad \forall i \in V_f, \forall k \in K \quad (53)$$

Constraint (53) guarantees that the defined routes are all linked in order to build a single rotation.

Since this formulation disregards the rotation's duration, duration constraints are assessed afterwards. If $\text{Dur} > T$, then the solution is discarded and the next pair of rotations from the list is chosen. If $\text{Dur} \leq T$, then the new rotation is part of the final solution. If the number of rotations is still larger than the number of vehicles available, module 4 is run again until the number of rotations is equal to the number of vehicles.

3.5 Post-Optimization

In this module, some post-optimization procedures are applied to the final solution. The 2-opt and 3-opt exchanges are applied within and between rotations, ensuring that an exchange is only accepted if the capacity and duration constraints are not violated. Since a rotation can include single- and/or inter-depot routes, exchanging edges of inter-depot routes might result in a single-depot route from another depot. Figure 4 illustrates this situation where rotation 4 (in red) is composed by two inter-depot routes and one single-depot route, with a vehicle based at depot 26. The two inter-depot routes have two crossing edges (27,20) and (13,21), and after applying the 2-opt operation, these routes turn into two single-depot routes: one from depot 27 [27-25-13-27] and the other from depot 26 [26-21-20-23-3-26]. This means that route [27-25-13-27] now belongs to rotation 1 (in blue), and the duration must be checked. In this case, including route [27-25-13-27] in rotation 1 makes it infeasible regarding the duration limit, and therefore, this move is not accepted, although it reduces the overall cost.

4 Computational results

In this section, the benchmark instances for the MDVRPI developed by Crevier et al. (2007) and Muter et al. (2014) are solved via the proposed methodology. Moreover, to show how the MDVRPI formulation scales, small instances are also solved through the full MDVRPI formulation using the branch-and-bound algorithm embedded in the CPLEX solver. Both MDVRPI full model and the first four modules of the solution methodology were implemented in GAMS 23.7 and solved through the CPLEX Optimizer 12.3.0, on an Intel Xeon CPU X5680 @ 3.33 GHz. The CPU time was limited to 4 h in each module. The fifth module was implemented in Python. Since the benchmark instances do not consider any fixed cost for routes,

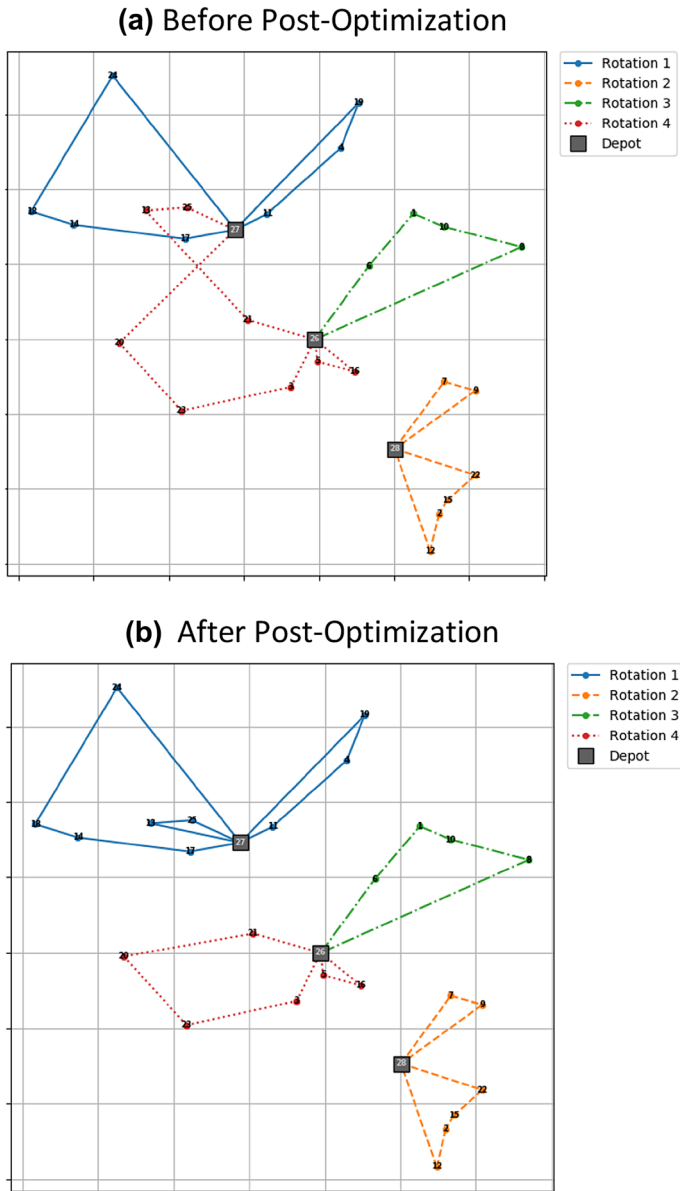


Fig. 4 Illustrative example of the post-optimization procedure (color figure online)

the fixed cost H is set to zero in module 1. In module 3, parameter B was set to 0.5 after performing some preliminary tests.

In the Muter et al. (2014) instances, the number of customers is limited to 25 and 40 and the number of vehicles is limited to four and six. In the Crevier et al. (2007) instances, the number of customers is between 48 and 288 and the number

Table 2 Comparison between the MDVRPI formulation with the matheuristic for small-size instances

Instances	$ V_j $	R	MDVRPI formulation@CPLEX				Solution methodology		
			LB	UB	Gap (%)	CPU	LB	UB	CPU
<i>a1-10-50-450</i>	3	2	336.248	336.248	0.0	43	336.248*	336.248	0.9
<i>a1-12-50-450</i>	3	2	354.315	354.315	0.0	112	354.315*	354.315	1.6
<i>a1-14-50-450</i>	3	4	401.899	401.899	0.0	6997	401.899*	401.899	1.8
<i>a1-16-50-450</i>	3	4	439.639	–	–	14,400	469.369*	469.369	2.6
<i>a1-18-50-450</i>	3	4	508.903	–	–	14,400	540.411*	540.411	5.7

*Solution proven to be optimal for module 1

of vehicles is between three and six. To test the MDVRPI formulation, we create small instances based on the original ones, with the number of customers ranging from 10 to 18, and the number of vehicles between two and four. We selected the first instance “*a1*” and considered the first n customers for this test. The results are shown in Table 2. In the first column, each instance is coded as in the Muter et al.’s (2014) work: $\#-n-Q-T$. The number of depots is shown in the second column and the number of vehicles in the third column. The following four columns present the results obtained when solving the MDVRPI formulation via CPLEX: lower bound (LB), upper bound (UP), gap between the upper and lower bound and the CPU time (in seconds). The last three columns are the results when solving the instances with the solution methodology: LB is the branch-and-bound lower bound of module 1 (solutions proven to be optimal for module 1 are asterisked in the LB column), UB is the final objective function value obtained after the whole procedure has been applied and the CPU time, in seconds, taken by running our method.

It can be observed from Table 2 that CPLEX is only capable of solving very-small-size instances of the MDVRPI (up to 14 customers) and takes much more CPU time to reach the same solution when compared to the proposed solution methodology. With 16 customers (*a1-16-50-450*), CPLEX failed to find a feasible solution after 4 CPU hours, while the proposed matheuristic found the optimal solution in 2.6 s.

Given the results of Table 2, the benchmarking instances of Crevier et al. (2007) and Muter et al. (2014) will be solved only by our solution methodology, and the results compared with the work of Muter et al. (2014)—see Table 3. The results are only compared with the work of Muter et al. (2014) since it is the only work that solved the instances as a MDVRPI (with vehicles based at multiple depots). Note that Crevier et al. (2007) solved the original instances as a VRPIF (with all vehicles based at a central depot) and not as a MDVRPI. Table 3 shows the results presented by Muter et al. (2014): lower bound (LB), upper bound (UB) and CPU time (in seconds), and the final three columns show our own results.

When analysing the results, we observe that our solution method provides a lower bound for all 66 instances, while the approach proposed by Muter et al. (2014) was not able to provide a lower bound for *i1-25-50-450* and *k1-25-50-450*, nor was it able to solve the original instances (those having more than 40 customers). Focusing

Table 3 Results for MDVRPI instances

Instances	V _j	R	Muter et al. (2014)			This work		
			LB	UB	CPU ^a	LB	UB	CPU
<i>a1-25-50-450</i>	3	4	691.699	693.810	1967	693.810*	693.810	19
<i>b1-25-50-450</i>	3	4	714.405	714.405	7	714.405*	738.358	20
<i>c1-25-50-450</i>	3	4	825.098	845.542	25,476	845.542*	896.236	63
<i>d1-25-50-450</i>	4	4	750.744			755.174*	763.303	25
<i>e1-25-50-450</i>	4	4	789.189	803.715	223	799.722*	814.839	26
<i>f1-25-50-450</i>	4	4	551.828	551.828	41	551.828*	551.828	27
<i>g1-25-50-450</i>	5	4	654.016	654.016	41	654.016*	654.016	49
<i>h1-25-50-450</i>	5	4	557.610	558.646	138	547.074*	558.646	1551
<i>i1-25-50-450</i>	5	4				816.085*	899.345	51
<i>j1-25-50-450</i>	6	4	769.605			731.072*		
<i>k1-25-50-450</i>	6	4				838.858*		
<i>l1-25-50-450</i>	6	4	818.280	818.280	746	809.191*	855.449	21
<i>a1-40-50-450</i>	3	6	991.657	998.431	29,278	998.431*	998.431	893
<i>b1-40-50-450</i>	3	6	1053.850	1059.370	4474	1059.370*	1059.370	2225
<i>c1-40-50-450</i>	3	6	1140.250			1148.669*	1148.669	12,037
<i>d1-40-50-450</i>	4	6	1026.770			1048.560*	1061.807	15,861
<i>e1-40-50-450</i>	4	6	1218.140	1236.620	22,334	1229.929*	1236.620	19,609
<i>f1-40-50-450</i>	4	6	831.828			854.108*	854.108	6151
<i>g1-40-50-450</i>	5	6	1022.740			1028.460*	1054.123	20,985
<i>h1-40-50-450</i>	5	6	870.594	875.552	10,893	874.030*	875.552	19,328
<i>i1-40-50-450</i>	5	6	1211.850	1222.840	27,712	1219.724*		
<i>j1-40-50-450</i>	6	6	890.826			888.219*		
<i>k1-40-50-450</i>	6	6	1227.980			1234.232*		
<i>l1-40-50-450</i>	6	6	1076.860	1085.320	35,555	1063.013*		
<i>a2-25-50-450</i>	5	4	708.377			716.137*	727.992	22
<i>b2-25-50-450</i>	5	4	906.089	912.429	433	888.493*		14,469
<i>c2-25-50-450</i>	5	4	676.411	683.188	4330	673.647*		
<i>d2-25-50-450</i>	5	4	864.477	876.113	1235	858.715*	890.189	14,451
<i>e2-25-50-450</i>	5	4	695.963			693.616*	701.609	60
<i>f2-25-50-450</i>	5	4	779.543	781.176	275	765.299*	823.758	14,428
<i>g2-25-50-450</i>	7	4	793.633	794.243	772	786.746*	793.633	19
<i>h2-25-50-450</i>	7	4	712.492	716.22	774	684.805*		
<i>i2-25-50-450</i>	7	4	909.809	910.505	1508	870.897*		
<i>j2-25-50-450</i>	7	4	609.378	609.378	447	604.955*	612.021	19
<i>a2-40-50-450</i>	5	6	1007.91	1010.61	5872	1008.381*		
<i>b2-40-50-450</i>	5	6	1234.26	1238.94	8881	1230.582*	1243.321	7919
<i>c2-40-50-450</i>	5	6	1142.89			1156.045*	1156.045	897
<i>d2-40-50-450</i>	5	6	1183.18			1196.819*	1203.305	2661
<i>e2-40-50-450</i>	5	6	1072.35			1081.201*	1092.262	11,230
<i>f2-40-50-450</i>	5	6	1023.46			1046.454*	1046.454	4693
<i>g2-40-50-450</i>	7	6	1033.81	1035.01	6895	1027.798*	1046.867	7688

Table 3 (continued)

Instances	$ V_j $	R	Muter et al. (2014)			This work		
			LB	UB	CPU ^a	LB	UB	CPU
<i>h2-40-50-450</i>	7	6	939.896	940.924	15,462	928.925*	973.971	17,346
<i>i2-40-50-450</i>	7	6	1155.96			1133.831*	<i>1200.142</i>	7818
<i>j2-40-50-450</i>	7	6	877.515			871.766*		
<i>a1-48-60-550</i>	3	6				1069.498	1102.575	14,400
<i>b1-96-210-1200</i>	3	4				1167.858	1203.977	14,400
<i>c1-192-360-1850</i>	3	5				1776.140	1825.427	14,400
<i>d1-48-80-600</i>	4	5				1014.828*	1020.148	17,030
<i>e1-96-230-1300</i>	4	5				1280.707*	1293.033	20,394
<i>f1-192-380-2000</i>	4	4				1508.436	1556.694	14,400
<i>g1-72-80-750</i>	5	5				1112.332	1151.669	28,801
<i>h1-144-230-1550</i>	5	4				1455.586	1520.988	14,400
<i>i1-216-380-2350</i>	5	4				1831.927	1908.257	14,400
<i>j1-72-100-800</i>	6	4				1035.688	1079.273	28,801
<i>k1-144-250-1650</i>	6	4				1475.089	1553.064	28,800
<i>l1-216-400-2500</i>	6	4				1780.184	1831.465	14,400
<i>a2-48-150-600</i>	5	4				893.001*	900.840	14,571
<i>b2-96-200-1150</i>	5	4				1227.089	1266.383	28,801
<i>c2-144-250-1700</i>	5	4				1621.010	1683.292	14,400
<i>d2-192-300-2250</i>	5	3				1763.021	1849.970	14,400
<i>e2-240-350-2800</i>	5	3				1839.061	1915.688	14,400
<i>f2-288-400-3350</i>	5	3				2175.805	2258.996	28,800
<i>g2-72-175-950</i>	7	4				1074.764	1112.352	28,801
<i>h2-144-250-1800</i>	7	4				1485.507	1539.386	14,400
<i>i2-216-325-2650</i>	7	3				1824.002	1927.953	28,800
<i>j2-288-400-3500</i>	7	3				2160.304	2272.526	28,800

*Solution proven to be optimal for module 1

^aThe CPU reported corresponds to the CPU reported in Muter et al.'s (2014) work, where a computer with a 2.67-GHz Intel Westmere-EP X5650 Processor and 4 GB of RAM was used

on the 25 and 40 customers instances, our matheuristic improves the lower bound in 20 of the 42 instances and the same lower bound is obtained for 3 other instances (in bold). Also an upper bound for 13 instances, not reported in Muter et al.'s (2014) work, was obtained (in bold italic), four of them (*c1-40-50-450*, *f1-40-50-450*, *c2-40-50-450* and *f2-40-50-450*) being proven to be optimal (in bold italic underlined). Furthermore, when analysing the results reported by Muter et al. (2014) we noticed some inconsistencies. For example, in instances *b1-25-50-450* and *c1-25-50-450*, our lower bound corresponds to the upper bound presented by Muter et al. (2014). From our analysis, we conclude that the Muter et al. (2014) solutions are unfeasible, as they require five vehicles and the maximum number allowed for those instances is four vehicles. We were able to reach a feasible solution since our approach reduces

the number of vehicles needed (by executing module 4). Additionally, for instance g2-25-50-450, we find a better feasible solution than Muter et al. (2014)—793.633 versus 794.243.

In total, our method solved 54 out of 66 instances, 22 of them from the original MDVRPI instances, which are solved, to our best knowledge, for the first time in this work, with vehicles located at different depots. The reasons behind why all larger instances were solved (original instances) but not all medium instances (25-customer and 40-customer) can be found by analysing the detailed results displayed in Tables 4, 5 and 6. For the 25-customer instances (Table 4), 10 out of 22 instances need to be worked out by module 3; for the 40-customer instances (Table 5), 15 out of 22 instances need to go through module 3; while for the original and larger instances (Table 6), only 7 out of 22 instances go to module 3. The performance of module 3 depends on the number of routes involved in a rotation [set U cardinality at Constraint (52)]. As mentioned, Constraint (52) is implemented in a cutting-plane fashion, i.e. we start with $|U| = 2$; if a feasible solution is obtained (all rotations with $Dur \leq T$), module 3 stops; otherwise, the cardinality of set U increases by one unit until a feasible solution is obtained. For example, when Constraint (52) is added with $|U| = 2$, it is guaranteed that a rotation with two inter-depot routes does not exceed the maximum duration allowed for any rotation. However, this does not prevent the appearance of a rotation with three or more inter-depot routes with a duration higher than T in the solution. In this case, we run module 3 again, now considering $|U| = 3$, and so on. We noticed that worse results were obtained as the cardinality of set U increases due to the exponential number of constraints generated. For the original (and larger) instances, the cardinality of set U was always, at most, two or three. For the smaller instances, given the location of the customers and the large number of depots (3–6 depots to serve only 25 or 40 customers), rotations with a higher number of inter-depot routes are created, linking several depots. Consequently, $|U|$ needed to be higher, and with $|U| \geq 5$ module 3 fails to provide a solution.

Given that the original MDVRPI had been solved in previous works as a VRPIF with all vehicles located at a central depot, we are able to assess the economic benefit of positioning vehicles at different depots by comparing our results with the best known solution (BKS) for each instance solved as a VRPIF. The BKS is available in the works of Crevier et al. (2007), Tarantilis et al. (2008) and Hemmelmayr et al. (2013). This comparison is shown in Table 7. We can observe that a potential benefit can reach values in the order of 10%. The potential benefit for the smaller instances, with fewer customers, is higher than that of larger instances. This fact is also observed in Markov et al. (2016), where the potential savings from a flexible assignment of destination depots and from home depot optimization are assessed. Comparing the average benefits gained: when vehicles are based at a central depot but are able to end their rotation at a different one, the average benefit is 1.77% (Markov et al. 2016); when vehicles can be based at different depots, but have to start and end their rotation at the same depot, the average benefit is 2.13% (Table 7); and finally, when vehicles may be based at different depots and can both start and end their route at different depots, the average benefit is 2.54% (Markov et al. 2016).

To illustrate the difference between a VRPIF and MDVRPI, instance $a1$ is analysed in further detail. Figure 5 presents both solutions for instance $a1$: the one

Table 4 Detailed results obtained on 25-customer MDVRPI instances

Instance	Module 1			Module 2			Dura- tion > T?	Module 3			Module 2 (after Mod.3)			Module 4			Mod- ule 5 Cost		
	Cost	CPU time (%)	GAP (%)	N. Rou.	N. Rot.	CPU time		N. Iter.	GAP (%)	N. Cust.	Cost	CPU time	GAP (%)	N. Rot.	CPU time	N. Cust.		Cost	N. Iter.
a1-25-50-450	693.810	18.7	0.0	8	4	0.4	1	0.0	No	-	-	-	-	-	-	-	-	-	-
b1-25-50-450	714.405	18.4	0.0	8	5	0.4	1	0.0	No	-	-	-	-	-	8	738.358	1.5	1	0.0
c1-25-50-450	845.542	61.4	0.0	8	5	0.3	1	0.0	No	-	-	-	-	-	6	925.051	1.3	1	0.0
d1-25-50-450	755.174	23.7	0.0	10	5	0.4	1	0.0	No	-	-	-	-	-	4	763.303	0.5	1	0.0
e1-25-50-450	799.722	24.4	0.0	7	5	0.3	1	0.0	No	-	-	-	-	-	7	814.839	1.2	1	0.0
f1-25-50-450	551.828	27.2	0.0	9	4	0.3	1	0.0	No	-	-	-	-	-	-	-	-	-	-
g1-25-50-450	654.016	48.6	0.0	8	4	0.1	1	0.0	No	-	-	-	-	-	-	-	-	-	-
h1-25-50-450	547.074	16.4	0.0	7	4	0.1	5	0.0	Yes	16	559.054	1534	0	4	0.3	0	Yes	-	-
i1-25-50-450	816.085	11.8	0.0	7	5	0.3	1	0.0	No	-	-	-	-	-	9	899.345	39.0	1	0.0
j1-25-50-450	731.072	10.9	0.0	9	5	0.1	14	0.0	Yes	21	*	*	*	*	-	-	-	-	-
k1-25-50-450	838.858	32.7	0.0	7	3	0.1	10	0.0	Yes	22	*	*	*	*	-	-	-	-	-
l1-25-50-450	809.191	20.4	0.0	9	6	0.3	1	0.0	No	-	-	-	-	-	11	855.449	0.6	2	0.0
m1-25-50-450	716.137	20.4	0.0	8	5	0.2	1	0.0	No	-	-	-	-	-	7	727.992	1.6	1	0.0
n1-25-50-450	888.493	49.4	0.0	10	5	0.2	4	0.0	Yes	11	972.364	14,400	4.7	6	0.147	0.0	Yes	10	Infeasible
o1-25-50-450	673.647	22.1	0.0	8	4	0.1	11	0.0	Yes	18	*	*	*	*	-	-	-	-	-
p1-25-50-450	858.715	48.2	0.0	9	4	0.4	10	0.0	Yes	16	861.981	14,400	6.0	5	0.4	0.0	Yes	8	890.189
q1-25-50-450	693.616	36.1	0.0	10	4	0.3	5	0.0	Yes	9	695.895	21.9	0.0	5	0.3	0.0	No	7	701.609
r1-25-50-450	765.299	25.7	0.0	8	5	0.2	4	0.0	Yes	12	809.083	14,400	5.4	5	0.2	0.0	Yes	8	823.758

Table 4 (continued)

Instance	Module 1			Module 2			Duration > T?		Module 3			Module 2 (after Mod.3)			Module 4		Module 5		
	Cost	CPU time (%)	GAP (%)	N. Rot.	N. Rot.	CPU time	N.Iter.	GAP (%)	N.Cust.	Cost	CPU time	GAP (%)	N.Rot. ≤ IRI?	N.Cust.	Cost	CPU time	N.Iter.	GAP (%)	Cost
<i>g2-25-50-450</i>	786.746	18.9	0.0	8	5	0.2	1	0.0	No	-	-	-	No	5	793.633	0.3	1	0.0	
<i>l2-25-50-450</i>	684.805	32.8	0.0	9	4	0.2	16	0.0	Yes	20	*	*							
<i>i2-25-50-450</i>	870.897	33.3	0.0	10	5	0.2	12	0.0	Yes	15	*	*							
<i>j2-25-50-450</i>	604.955	17.5	0.0	9	5	0.2	1	0.0	No	-	-	-	No	7	612.021	1.2	1	0.0	

GAP percentage deviation between the upper bound and the lower bound in the branch-and-bound algorithm of the CPLEX after the time limit has been reached
N.Rot. number of routes, *N.Rot.* number of rotations, *N.Iter.* number of iterations, *N.Cust.* number of customers
 *Module 3 failed to obtain a solution

obtained by solving the case as a MDVRPI (on the left) and as a VRPIF (on the right).

In the VRPIF solution, twelve routes are defined involving five vehicle rotations that must be based out of depot 49. In the MDVRPI, twelve routes are also defined and assigned to five vehicle rotations that are based out of the three depots (this solution involves two vehicles at depot 51, one vehicle at depot 49 and two vehicles at depot 50), representing a cost saving of 6.5%.

In order to illustrate how the solution method works, Figs. 6, 7, 8, 9 and 10 display the solutions generated by each module for instance *a2*. This instance was chosen since its resolution requires the execution of all modules of the solution method.

The solution obtained by module 1 is comprised of seven routes as illustrated in Fig. 6.

These seven routes are the input for module 2 where the rotations are defined. In the first iteration, the total duration for a rotation (Dur) is set to 600 (T). Since no feasible solution was obtained, a second iteration was performed with Dur value reset to 650 ($T + \Delta$) with Δ as the minimum duration among the seven routes (i.e. 50 for route 5). A feasible solution was obtained in the fourth iteration, with $Dur = 750$, defining five rotations (see Fig. 7).

The total duration of rotation 1 is 721, rotation 2 is 54, rotation 3 is 299, rotation 4 is 478 and rotation 5 is 164. Since $T = 600$, rotation 1 needs to be re-worked by module 3, as it exceeds the operational time limit. Rotations 2, 3 and 5 have a duration lower than half of 600, meaning that they are possible candidates to be added to module 3 (since $B = 0.5$). However, the capacity usage of rotation 3 is 0.93 excluding this rotation from being re-worked by module 3, as it is above the assumed minimum capacity usage. The other two rotations (2 and 5) have a capacity usage lower than $B = 0.5$, and therefore, the customers in rotations 1, 2 and 5 (in this case, 27 customers) are considered as input for module 3. The solution obtained in module 3 for the 27 customers is shown in Fig. 8, where we obtained two single-depot and two inter-depot routes satisfying the duration limit.

After running module 3, module 2 is executed once more with the four new routes created by module 3 and the three previous routes from rotations 3 (with one route) and 4 (with two routes). Five rotations were obtained, meaning that the maximum number of vehicles available was exceeded ($|R| = 4$) (see Fig. 9). Therefore, module 4 was executed.

Firstly, module 4 elects the pair of rotations to be merged. Given the maximum duration for a rotation ($T = 600$), four pairs of rotation could be merged: [1; 2], [2; 3], [2; 4] and [2; 5]. For each pair, the minimum distance between each rotation is assessed considering both customers and depots and the pair with the lowest value is chosen. In instance *a2*, the pair with the minimum distance value is [2; 5]. The sixteen customers belonging to this pair are the input for the mathematical formulation of module 4 in order to define one single rotation. The solution obtained in module 4 is given in Fig. 10. The post-optimization procedures (module 5) were not able to improve the solution of instance *a2*, so the solution obtained at module 4 is the final one. This solution is to base the four vehicles out of facilities 49, 51, 52 and 53 (each represented by a different type of line). Therefore, these facilities act as depots and facility 50 remains unused.

Table 5 Detailed results obtained on 40-customer MDVRPI instances

Instance	Module 1			Module 2			Dura- tion > 7?	Module 3			Module 2 (after Mod.:3)			N. Rot. ≤ IRP?	Module 4		Mod- ule 5 Cost
	Cost	CPU time	GAP (%)	N. Rou.	CPU Rot. time	N. Iter.		N. Cust.	Cost	CPU time	GAP (%)	N. Rot.	CPU Rot. time		GAP (%)	N. Cust.	
a1-40-50-450	998.431	893	0.0	13	6	0.6	1	0.0	No	-	-	-	-	Yes	-	-	-
b1-40-50-450	1059.370	2225	0.0	12	5	0.3	1	0.0	No	-	-	-	-	Yes	-	-	-
c1-40-50-450	1148.669	12,036	0.0	13	6	0.4	1	0.0	No	-	-	-	-	Yes	-	-	-
d1-40-50-450	1048.560	1721	0.0	13	5	0.2	6	0.0	Yes	15	1061.807	14,130	0	6	0.3	0.0	-
e1-40-50-450	1229.929	5208	0.0	12	5	0.4	4	0.0	Yes	21	1236.619	14,400	3.8	6	0.3	0.0	-
f1-40-50-450	854.108	6150	0.0	14	4	0.3	1	0.0	No	-	-	-	-	Yes	-	-	-
g1-40-50-450	1028.460	6584	0.0	14	5	0.3	12	0.0	Yes	16	1054.123	14,400	4.5	6	0.2	0.0	-
h1-40-50-450	874.030	10,627	0.0	12	5	0.4	8	0.0	Yes	14	875.552	8700	0	6	0.3	0.0	-
i1-40-50-450	1219.724	5313	0.0	12	5	0.3	8	0.0	Yes	19	*	*	*	-	-	-	-
j1-40-50-450	888.219	1365	0.0	10	4	0.2	11	0.0	Yes	20	*	*	*	-	-	-	-
k1-40-50-450	1234.232	14,390	0.0	13	6	0.2	11	0.0	Yes	27	*	*	*	-	-	-	-
l1-40-50-450	1063.013	1001	0.0	14	7	0.2	14	0.0	Yes	26	*	*	*	-	-	-	-
a2-40-50-450	1008.381	6087	0.0	13	5	0.4	7	0.0	Yes	20	*	*	*	-	-	-	-
b2-40-50-450	1230.582	693	0.0	12	6	0.3	2	0.0	Yes	7	1243.321	7225	0.0	6	0.3	0.0	-
c2-40-50-450	1156.045	897	0.0	14	6	0.4	1	0.0	No	-	-	-	-	Yes	-	-	-
d2-40-50-450	1196.819	1694	0.0	12	5	0.4	3	0.0	Yes	16	1203.305	966	0	6	0.3	0.0	-
e2-40-50-450	1081.201	10,837	0.0	15	6	0.3	4	0.0	Yes	12	1091.354	390	0	7	0.4	0.0	1
f2-40-50-450	1046.454	4693	0.0	13	5	0.3	1	0.0	No	-	-	-	-	Yes	-	-	-

Table 5 (continued)

Instance	Module 1			Module 2			Dura- tion > T?			Module 3			Module 2 (after Mod.3)			Module 4			Mod- ule 5				
	Cost	CPU time	GAP (%)	N. Rou.	N. Rot.	CPU time	N. Iter.	GAP (%)	N. Iter.	Dura- tion > T?	N. Cust.	Cost	CPU time	GAP (%)	N. Rot.	CPU time	GAP (%)	N. Cust.	Cost	CPU time	N. Iter.	GAP (%)	Cost
g2-40-50-450	1027.798	7630	0.0	14	7	0.2	1	0.0	No	-	-	-	-	-	No	-	-	8	1070.256	58	1	0.0	1046.867
h2-40-50-450	928.925	2943	0.0	14	7	0.2	11	0.0	Yes	20	947.932	14.400	10	8	0.4	0.0	0.0	14	973.971	1.9	2	0.0	0.0
i2-40-50-450	1133.831	3435	0.0	12	6	0.4	11	0.0	Yes	14	1159.64	4380	0	8	0.4	0.0	0.0	18	1200.142	2.4	2	0.0	0.0
j2-40-50-450	871.766	6172	0.0	14	6	0.3	10	0.0	Yes	19	*	*	*	*	*	*	*						

GAP percentage deviation between the upper bound and the lower bound in the branch-and-bound algorithm of the CPLEX after the time limit has been reached

N.Rou, number of routes, N.Rot, number of rotations, N.Iter, number of iterations, N.Cust, number of customers

*Module 3 failed to obtain a solution

Table 6 Detailed results obtained on original MDVRPI instances

Instance	Module 1				Module 2				Duration > T?	Module 3				Module 2 (after Mod.3)				N. Rot. ≤ IR?	Module 4				Module 5	
	Cost	CPU time	GAP (%)	N. Rou.	N. Rot.	CPU time	N. Iter.	GAP (%)		N. Cust.	Cost	CPU time	GAP (%)	N. Cust.	Cost	CPU time	GAP (%)		N. Cust.	Cost	CPU time	N. Iter.		GAP (%)
a1-48-60-550	1102.58	14,400	3.0	12	5	0.2	1	0.0	No	-	-	-	-	-	-	-	-	-	-	-	-	-		
b1-96-210-1200	1203.98	14,400	3.0	7	4	0.2	1	0.0	No	-	-	-	-	-	-	-	-	-	-	-	-	-		
c1-192-360-1850	1825.43	14,400	2.7	8	5	0.3	1	0.0	No	-	-	-	-	-	-	-	-	-	-	-	-	-		
d1-48-80-600	1014.83	2629	0.0	10	5	1	4	0.0	Yes	26	1020.15	14,400	12.0	5	0.3	0.0	Yes	-	-	-	-	-	-	
e1-96-230-1300	1280.71	5993	0.0	6	2	0.8	3	0.0	Yes	52	1293.03	14,400	6.5	3	0.3	0.0	Yes	-	-	-	-	-	-	
f1-192-380-2000	1556.69	14,400	3.1	9	4	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
g1-72-80-750	1127.68	14,400	2.7	15	4	0.9	3	0.0	Yes	41	1151.67	14,400	13.7	4	0.3	0.0	Yes	-	-	-	-	-	-	
h1-144-230-1550	1520.99	14,400	4.3	9	4	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
i1-216-380-2350	1908.26	14,400	4.0	9	3	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
j1-72-100-800	1066.62	14,400	2.9	10	4	0.5	2	0.0	Yes	33	1079.27	14,400	9.3	4	0.3	0.0	Yes	-	-	-	-	-	-	-
k1-144-250-1650	1546.22	14,400	4.6	10	5	0.3	1	0.0	No	-	-	-	-	-	-	-	No	55	1554.05	14,400	1	4.6	1553.06	
l1-216-400-2500	1831.46	14,400	2.8	9	4	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
m1-48-150-600	893.00	146	0.0	7	5	1	4	0.0	Yes	27	899.00	14,400	6.0	5	0.2	0.0	No	16	900.84	24	1	0.0	-	
n1-96-200-1150	1250.86	14,400	1.9	7	3	1.4	5	0.0	Yes	57	1266.38	14,400	3.9	4	0.3	0.0	Yes	-	-	-	-	-	-	-
o1-144-250-1700	1683.29	14,400	3.7	9	4	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
p1-192-300-2250	1849.97	14,400	4.7	9	3	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
q1-240-350-2800	1915.69	14,400	4.0	10	3	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	
r1-288-400-3350	2257.06	14,400	3.6	11	4	0.3	1	0.0	No	-	-	-	-	-	-	-	No	98	2259.00	14,400	1	3.8	-	

Table 6 (continued)

Instance	Module 1			Module 2			Dura- tion > T?		Module 3			Module 2 (after Mod.3)			Module 4		Mod- ule 5				
	Cost	CPU time	GAP (%)	N. Rou.	N.	CPU time	N. Iter.	GAP (%)	N. Cust.	Cost	CPU time	GAP (%)	N. Rot.	CPU time	GAP (%)	N. Cust.	Cost	N. Iter.	GAP (%)	Cost	
g2-72-175-950	1080.16	14,400	0.5	6	3	1.3	5	0.0	Yes	55	1116.44	14,400	9.5	4	0.3	0.0	Yes	-	-	-	1112.35
h2-144-250-1800	1539.39	14,400	3.5	10	4	0.3	1	0.0	No	-	-	-	-	-	-	-	Yes	-	-	-	-
i2-216-325-2650	1920.00	14,400	5.0	10	4	0.4	1	0.0	No	-	-	-	-	-	-	-	No	96	1927.95	14,400	4.8
j2-288-400-3500	2266.85	14,400	4.7	11	5	0.4	1	0.0	No	-	-	-	-	-	-	-	No	225	2272.53	14,400	4.5

GAP percentage deviation between the upper bound and the lower bound in the branch-and-bound algorithm of the CPLEX after the time limit has been reached
N.Rou. number of routes, *N.Rot.* number of rotations, *N.Iter.* number of iterations, *N.Cust.* number of customers

Table 7 Potential benefit from positioning vehicles at different depots

Instance	Best known solution (VRPIF)	This work (MDVRPI)	Deviation (%)
<i>a1</i> -48-60-550	1179.79	1102.575	-6.54
<i>b1</i> -96-210-1200	1217.07	1203.977	-1.08
<i>c1</i> -192-360-1850	1866.76	1825.427	-2.21
<i>d1</i> -48-80-600	1059.43	1020.148	-3.71
<i>e1</i> -96-230-1300	1309.12	1293.033	-1.23
<i>f1</i> -192-380-2000	1570.41	1556.694	-0.87
<i>g1</i> -72-80-750	1181.13	1151.669	-2.49
<i>h1</i> -144-230-1550	1545.50	1520.988	-1.59
<i>i1</i> -216-380-2350	1922.18	1908.257	-0.72
<i>j1</i> -72-100-800	1115.78	1079.273	-3.27
<i>k1</i> -144-250-1650	1576.36	1553.064	-1.48
<i>l1</i> -216-400-2500	1863.28	1831.465	-1.71
<i>a2</i> -48-150-600	997.94	900.840	-9.73
<i>b2</i> -96-200-1150	1291.19	1266.383	-1.92
<i>c2</i> -144-250-1700	1715.60	1683.292	-1.88
<i>d2</i> -192-300-2250	1856.84	1849.970	-0.37
<i>e2</i> -240-350-2800	1919.38	1915.688	-0.19
<i>f2</i> -288-400-3350	2230.32	2258.996	0.00
<i>g2</i> -72-175-950	1152.92	1112.352	-3.52
<i>h2</i> -144-250-1800	1575.28	1539.386	-2.28
<i>i2</i> -216-325-2650	1919.74	1927.953	0.00
<i>j2</i> -288-400-3500	2247.70	2272.526	0.00
Average			-2.13

Deviation was computed to 0% for instances *f2*, *i2* and *j2* since in the worst case, the results for the MDVRPI are equal to the results for the VRPIF

All the details of the final solutions produced by our approach for the original MDVRPI instances are available in supplementary material: Appendix B.

5 Conclusions

This paper addresses the MDVRPI, a routing problem characterized by the existence of multiple depots, where vehicles can reset their capacity without having to return to their home depot. This problem, which is very important in real logistic networks with applications in grocery distribution, waste collection, or when an electric vehicle fleet is present, has received little attention from academia.

This gap is explored in this work where a generic MDVRPI is studied considering that vehicles are based at multiple depots. A new formulation, based on the two-commodity flow formulation, was developed, where routes and rotations are defined in order to minimize the total routing cost. The location of the available vehicle

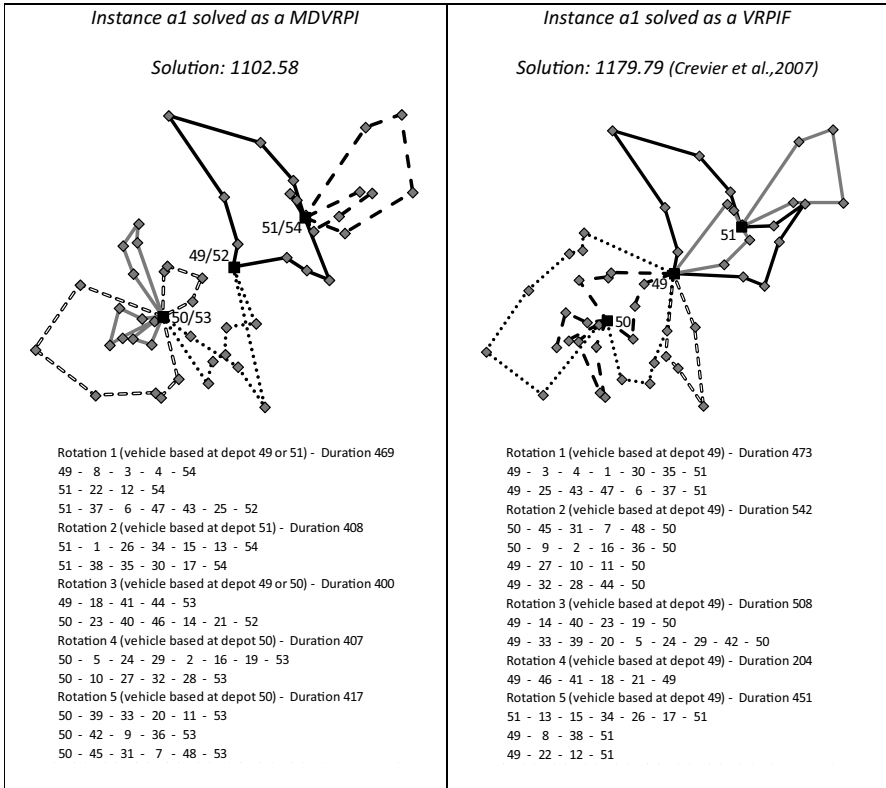


Fig. 5 Solution for instance *a1* solved as a MDVRPI and a VRPIF

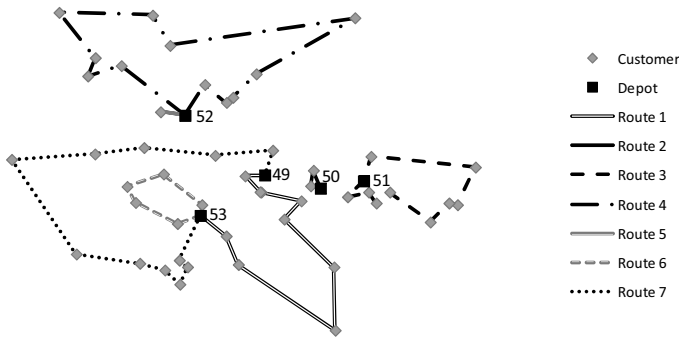


Fig. 6 Module 1 solution for instance *a2*

fleet as well as the role of each facility in the network (depot, intermediate facility, or both) is also decisions made by the model. Furthermore, to allow the solution of real instances and consequently potentiate model applicability to real cases, a matheuristic approach was developed. Literature instances were solved, namely

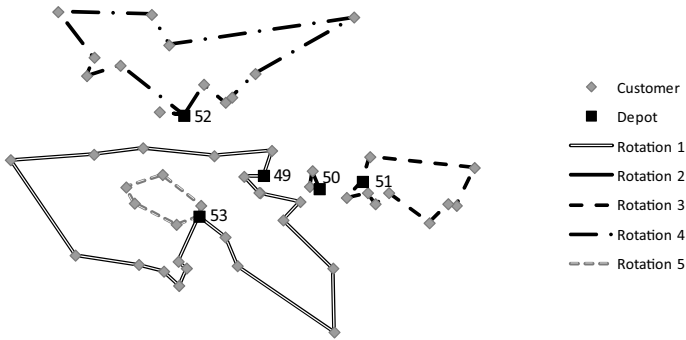


Fig. 7 Module 2 solution for instance *a2*

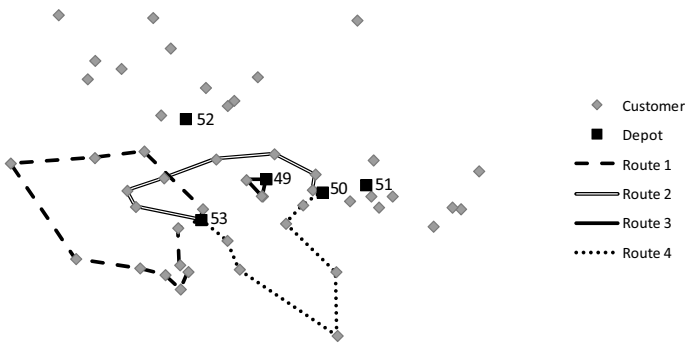


Fig. 8 Module 3 solution for instance *a2*

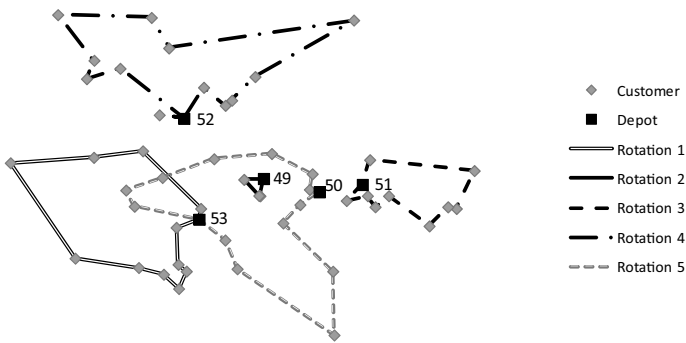


Fig. 9 Module 2 solution (after module 3) for instance *a2*

those proposed by Muter et al. (2014) and Crevier et al. (2007)—two reference works in the field. For the instances proposed in the first work, some new and better solutions were achieved, whereas instances proposed in the latter were solved for the first time as a MDVRPI. When compared to the VRPIF solutions reported by

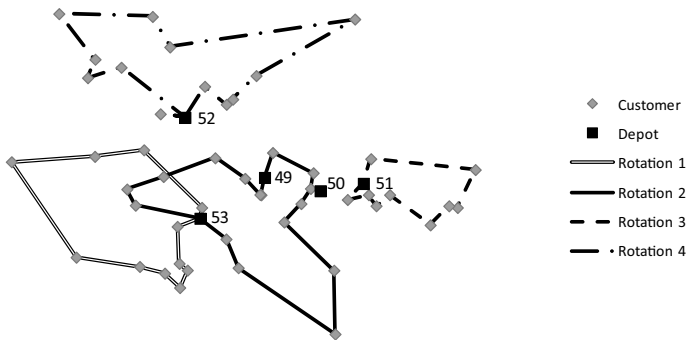


Fig. 10 Module 4 solution for instance a_2 (and the final one)

Crevier et al. (2007), Tarantilis et al. (2008) and Hemmelmayr et al. (2013), economic savings up to 10% can be observed using the proposed approach.

Finally, although important results were achieved in the present work, future research directions can also be identified. Namely, further work on solution methods should be pursued, allowing the solution of all benchmarking instances published in the literature, as well as real problems. Moreover, other types of problems based on the MDVRPI should also be explored. One example is the problem that combines the MDVRPI characteristics with the inventory management aspects. Such studies will enable the consideration of multiple products' availability at each facility. Also, holding costs could be accounted for into the MDVRPI problem together with routing costs, so as to model more realistic situations. Finally, the existence of uncertainty should also be explored within such problems.

Acknowledgements Tânia Ramos and Ana Barbosa-Póvoa acknowledge the support provided by FCT and P2020 under the project PTDC/EGE-OGE/28071/2017, Lisboa-01.0145-Feder-28071. Maria Isabel Gomes acknowledges the support provided by FCT under the project UID/MAT/00297/2019 (Centro de Matemática e Aplicações).

References

- Angelelli E, Speranza MG (2002) The periodic vehicle routing problem with intermediate facilities. *Eur J Oper Res* 137(2):233–247
- Baldacci R, Mingozzi A (2009) A unified exact method for solving different classes of vehicle routing problems. *Math Program* 120(2):347–380
- Baldacci R, Hadjiconstantinou E, Mingozzi A (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Oper Res* 52(5):723–738
- Bard JF, Huang L, Dror M, Jaillet P (1998) A branch and cut algorithm for the VRP with satellite facilities. *IIE Trans* 30(9):821–834
- Benjamin AM, Beasley JE (2010) Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Comput Oper Res* 37(12):2270–2280
- Brandao J, Mercer A (1997) A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *Eur J Oper Res* 100(1):180–191
- Cattaruzza D, Absi N, Feillet D, Vidal T (2014) A memetic algorithm for the multi trip vehicle routing problem. *Eur J Oper Res* 236(3):833–848

- Chao I, Golden BL, Wasil E (1993) A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. *Am J Math Manag Sci* 13:371–406
- Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Oper Res* 12(4):568–581
- Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* 30(2):105–119
- Crevier B, Cordeau JF, Laporte G (2007) The multi-depot vehicle routing problem with inter-depot routes. *Eur J Oper Res* 176(2):756–773
- Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *J Oper Res Soc Am* 2:393–410
- Dondo RGDGR, Cerda J (2009) A hybrid local improvement algorithm for large-scale multi-depot vehicle routing problems with time windows. *Comput Chem Eng* 33(2):513–530
- Gillet BE, Miller LR (1974) A Heuristic algorithm for the vehicle-dispatch problem. *Oper Res* 22:340–349
- Hemmelmayr V, Doerner K F, Hartl RF, Rath S (2013) A heuristic solution method for node routing based solid waste collection problems. *J Heuristics* 19(2):129–156
- Kim BI, Kim S, Sahoo S (2006) Waste collection vehicle routing problem with time windows. *Comput Oper Res* 33(12):3624–3642
- Laporte G (2009) Fifty years of vehicle routing. *Transp Sci* 43(4):408–416
- Laporte G, Semet F (2002) Classical heuristics for the capacitated VRP. In: Toth P, Vigo D (eds) *The vehicle routing problem, monographs on discrete mathematical and applications*. SIAM, Philadelphia, pp 109–128
- Laporte G, Nobert Y, Arpin D (1984) Optimal solutions to capacitated multi-depot vehicle routing problems. *Congressus Numerantium* 44:283–292
- Laporte G, Nobert Y, Taillefer S (1988) Solving a family of multi-depot vehicle-routing and location-routing problems. *Transp Sci* 22(3):161–172
- Lim A, Wang F (2005) Multi-depot vehicle routing problem: a one-stage approach. *IEEE Trans Autom Sci Eng* 2(4):397–402
- Lin S (1965) Computer solutions of the traveling salesman problem. *Bell Syst Tech J* 44(10):2245–2269
- Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. *Oper Res* 21:498–516
- Maniezzo V, Stützle T, Voß S (eds) (2009) *Matheuristics: hybridizing metaheuristics and mathematical programming*. *Annals of information system*. Springer, Berlin
- Markov I, Varone S, Bierlaire M (2016) Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. *Transp Res B* 84:256–273
- Mingozzi A, Roberti R, Toth P (2013) An exact algorithm for the multitrip vehicle routing problem. *Inf J Comput* 25(2):193–207
- Muter I, Cordeau JF, Laporte G (2014) A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transp Sci* 48(3):425–441
- Petch RJ, Salhi S (2003) A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Appl Math* 133:69–92
- Renaud J, Boctor FF, Laporte G (1996a) An improved petal heuristic for the vehicle routing problem. *J Oper Res Soc* 47:329–336
- Renaud J, Laporte G, Boctor FF (1996b) A tabu search heuristic for the multi-depot vehicle routing problem. *Comput Oper Res* 23(3):229–235
- Schneider M, Stenger A, Hof J (2015) An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectr* 37:353–387
- Tarantilis CD, Zachariadis EE, Kiranoudis CT (2008) A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *Inf J Comput* 20(1):154–168
- Toth P, Vigo D (eds) (2014) *Vehicle routing: problems, methods, and applications*, second edition. *MOS-SIAM series on optimization*, no. 18. SIAM, Philadelphia
- Tu W, Fang Z, Li Q, Shaw S-L, Chen B (2014) A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem. *Transp Res E* 61:84–97
- Van Breedam A (2001) Comparing descent heuristics and metaheuristics for the vehicle routing problem. *Comput Oper Res* 28(4):289–315
- Vidal T, Crainic T, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Oper Res* 60(3):611–624

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.