Wake-Up Oscillators with pW Power Consumption in Dynamic Leakage Suppression Logic

Optimization Letters
https://doi.org/10.1007/s11590-021-01845-7

ORIGINAL PAPER

The multi-depot *k*-traveling repairman problem

Maria Elena Bruni[1] · Sara Khodaparasti[1] · Iris Martínez-Salazar[2] · Samuel Nucamendi-Guillén[3]

Received: 6 July 2021 / Accepted: 10 December 2021

Abstract
In this paper, we study the multi-depot *k*-traveling repairman problem. This problem extends the traditional traveling repairman problem to the multi-depot case. Its objective, similar to the single depot variant, is the minimization of the sum of the arrival times to customers. We propose two distinct formulations to model the problem, obtained on layered graphs. In order to find feasible solutions for the largest instances, we propose a hybrid genetic algorithm where initial solutions are built using a splitting heuristic and a local search is embedded into the genetic algorithm. The efficiency of the mathematical formulations and of the solution approach are investigated through computational experiments. The proposed models are scalable enough to solve instances up to 240 customers.

Keywords Multi-depot *k*-traveling repairmen problem · Latency · Genetic algorithm

✉ Sara Khodaparasti
sara.khodaparasti@unical.it

Maria Elena Bruni
mariaelena.bruni@unical.it

Iris Martínez-Salazar
iris.martinezslz@uanl.edu.mx

Samuel Nucamendi-Guillén
snucamendi@up.edu.mx

[1] Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende, Cosenza, Italy

[2] Graduated Program in Systems Engineering, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Nuevo León, Mexico

[3] Escuela de Ciencias Económicas y Empresariales, Universidad Panamericana, Zapopan, Jalisco, Mexico

Published online: 04 February 2022

1 Introduction

In recent years, among practitioners and researchers, there has been a growing concern about the investigation of different problems arising in the class of Traveling Repairman Problem (TRP). Originally proposed in the repair and maintenance service context, this problem class has recently received attention from practitioners and researchers, stimulated by the wide range of applications of the problem in fields where the minimization of the sum of arrival times to customers is more important than other objective functions. Routing problems for emergency operations and post-disaster relief activities [4], home delivery services [23] and disk-head scheduling [3] and other customer-centered problems are examples of elective applicative fields.

The Multi-Depot k-TRP (MDk-TRP) generalizes the traditional single depot TRP to the multi-depot case and consists in minimizing the sum of the arrival times to the customers visited by a homogeneous fleet of k vehicles, housed at different depots. The introduction of cumulative costs into the objective function (i.e. the arrival time instead of the travel time as common in most routing problems) poses severe computational challenges [28] and the presence of multiple depots makes the problem even more complicated.

There is a lack of contributions on this problem variant, when compared to the standard multi-depot vehicle routing problem [25], even though it better reflects many real-world logistics systems. In the field of city logistics, for instance, widespread customer positions coupled with the sustainability challenge compelled the use of multiple distribution centers (also called satellites) [30]. This two-tier system is advantageous since each customer can be visited by vehicles from the closest satellite, reducing traffic jams and decreasing the level of emissions of associated pollutants.

The characteristics of spread customer positions, and multiple distribution centers is not only related to the last mile distribution system. Also in the health-care context, especially from the perspective of the design and optimization of emergency logistic systems in the aftermath of a disaster, relevant problems have been often modeled as a multi-depot routing problem [40,43]. Disasters can be devastating to communities and governments, and decision-making, in this customer-centric setting, needs to be quick and efficient since it becomes crucial to plan life-saving relief plans. There are numerous areas of disaster relief management for which the benefits of using multiple distributions centers are valuable. Timely delivery of medical supplies or reliefs, post-disaster assessment and recovery, are all linked to effective, customer-oriented multi-depot routing problems [45].

In this paper, we present two mathematical formulations for the MDk-TRP and we develop a genetic algorithm. Extensive computational experiments, run on a set of benchmark instances, show the efficiency of the proposed approach. The remainder of this paper is organized as follows. In Sect. 2, we provide a detailed review of the relevant literature. In Sect. 3, we describe the problem and present two new mathematical formulations using the concept of layered network. In Sect. 4, we explain the proposed genetic algorithm in detail. Section 5 presents the computational results carried out on a set of benchmark instances. Finally, our major conclusions and future research directions are presented in Sect. 6.

2 Related works

The TRP, also known as the minimum latency problem or the delivery man problem, has been extensively studied by a large number of researchers who proposed several exact and non-exact approaches. In [20] and [2] exact enumerative algorithms were developed, in which lower bounds are derived using a Lagrangian relaxation. An enumerative algorithm that makes use of lower bounds obtained from a linear integer programming formulation was proposed in [9]. A metaheuristic method which is based on a Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neighborhood Search (VNS) was proposed in [37]. A general VNS metaheuristic [24], was able to outperform the one suggested by [37].

The generalization of the TRP to the case with multiple identical vehicles (k-TRP) was first presented in [8]. In [21], the authors proposed an exact branch-and-price-and-cut algorithm for the multi-vehicle TRP with a distance constraint. An ad hoc label-setting algorithm with bi-directional search strategy is developed to solve the pricing problem. They tested 180 instances with up to 50 nodes and obtained an average gap less than 0.5%. In [28], a new and efficient mathematical formulation for the k-TRP, defined on a multi-layer network, was presented, then solved by an iterative greedy metaheuristic approach. In [1], five mathematical models for the multi-vehicle TRP are introduced. The first three models are derived from the classical and flow-based formulations, while the last two ones are generalizations of the time-dependent TRP formulation. The authors tested the formulations with instances up to 80 nodes and 16 vehicles, where the largest instance is solved in 86 s. The authors showed that the last two models perform better in terms of computational time. Interesting variants of k-TRP with service level as well as with profits or under uncertainty on travel times have been recently investigated [4,5].

A generalization of the TRP, which considers a demand associated to each node and adds capacity constraints is known as the Cumulative Capacitated Vehicle Routing Problem (CCVRP) [17]. In the last decade, a flourishing literature stream has been dedicated to the study of mathematical models [36], exact algorithms [22], as well as heuristic and metaheuristic approaches [18,29,35] for the class of CCVRPs. The investigation of heuristic and metaheuristics approaches for the CCVRP was conducted in [26] where a memetic heuristic procedure was developed; an adaptive large neighbourhood metaheuristic was proposed in [34]. In [39], the authors designed a hybrid metaheuristic algorithm for the min-max CCVRP enhanced with a two-stage adaptive variable neighbourhood search. An exact column generation approach was developed in [12], and later, the same authors proposed a heuristic approach in [13]. In [27], the authors introduced two new mathematical models for the CCVRP and proposed an iterated greedy algorithm outperforming other existing methods in the literature. Recently, in [11] was studied the CCVRP under uncertainty of demands, for which an approximation algorithm was developed.

The MDk-TRP is a variant of k-TRPs which considers the presence of multiple depots. Despite the richness of contributions for the multi-depot vehicle routing problem (interested readers are referred to [25,33,38] and references therein), a few papers deal with multi-depot variant of the TRP. We cite [19], where a three-index mathematical model is proposed for the multi-depot CCVRP. The authors also proposed a

partial optimization metaheuristic that decomposes the problem into sub-problems, later solved using either an approximate or an exact approach. To overcome the computational challenge posed by large-scale multi-depot CCVRP instances, a simple and efficient heuristic was proposed in [44] able to outperform the heuristic presented in [19] in terms of solution quality, computational time, as well as stability.

3 Mathematical models

The MDk-TRP is defined on an undirected graph $G = (V, E)$ where $V = C \cup D$ represents the vertex set composed of the set of customers C of cardinality n and the set of potential depots D of cardinality m. The set E denotes the edge set. Each edge $(i, j) \in E$ has associated a travel time t_{ij} for which the triangular inequality holds. According to the application at hand, this traveling cost can be interpreted as distance, time, fuel consumption, etc. A homogeneous fleet of k vehicles is dispatched from the m potential depots to visit all the n customers. Decisions also entail the selection of a subset of depots to host the vehicles and the determination of the number of vehicles to be sited at each active depot. The objective is to find a feasible routing plan such that the sum of the arrival times is minimized. Since the objective function is defined as the total latency (total waiting time), the travel time between the last visited customer by the vehicle and the depot to come back is not concerned. Therefore, we apply the terms tour and path (route) interchangeably throughout the paper. Strong mathematical models for the k-TRP [28] use a multi-layer network representation. The layered network is described as follows.

Let L be the set of levels $L = \{1, \ldots, r, \ldots, N\}$, where $N = n - k + 1$, and each level includes a copy of all the customers amended also with the depot in levels from 2 to N. Each path ends in the first level and starts in a copy of the depot in some level. In fact, the level number represents the position of the customer: the customer in the first level is the last one, the customer in the second level is the last but one, and so on. By using this network, it is easy to enforce the restriction that two distinct vehicles cannot visit the same customer, neither in the same level nor in different levels. In the next section, we will present two variants of the layered network enabling two new mathematical formulations for the MDk-TRP.

3.1 First formulation

Hereafter, we present the first mathematical formulation for the MDk-TRP. To extend the network to the multi-depot context, we add copies of the depots from levels 2 to N. Figure 1 illustrates this approach.

In Fig. 1, customers are displayed by squares and depots by circles. Node 0 denotes an auxiliary depot to specify the start of each path. As it can be seen, for each level, all the copies of the depots in the set D are linked to the corresponding starting node 0 at the same level. In other words, when the node 0 is active, one of the depots must be associated to it indicating the start of the path.

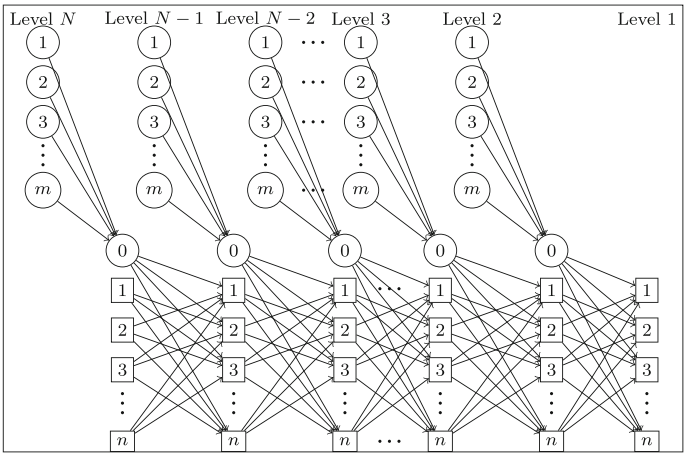

Fig. 1 Layered network for the MDk-TRP: first graphical representation

The variables are defined as follows. Let x_i^r be a binary variable that takes value 1 iff customer $i \in C$ is visited at level $r \in L$. If $x_i^r = 1$, customer i is active at level r and there are $r-1$ customers to be visited after in the same route. Let y_{ij}^r be another binary variable that is set to 1 iff $j \in C$ is visited right after $i \in C \cup \{0\}$ and there are exactly r customers to be visited after node i. Finally, w_{ij}^r represents a binary variable that takes value 1 iff $j \in C$ is the first customer to be visited (at level r, $r \geq 2$) by a vehicle dispatched from the depot $i \in D$ and there are r customers to be visited after. In other words, $w_{ij}^r = 1$ only if $y_{0j}^r = 1$, and 0 otherwise.
The first mathematical model is expressed as follows.

$$\min: \ z_1 = \sum_{i \in D} \sum_{j \in C} \sum_{r \in L} r\, t_{ij} w_{ij}^r + \sum_{i \in C} \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{r \in L \\ r \neq N}} r t_{ij} y_{ij}^r \tag{1}$$

$$\sum_{r \in L} x_i^r = 1 \ \ i \in C \tag{2}$$

$$\sum_{r \in L} \sum_{j \in C} y_{0j}^r = k \tag{3}$$

$$\sum_{i \in C} x_i^1 = \sum_{r \in L} \sum_{j \in C} y_{0j}^r \tag{4}$$

$$\sum_{\substack{j \in C \\ j \neq i}} y_{ij}^r = x_i^{r+1} \ \ i \in C,\ r \in L,\ r \neq N \tag{5}$$

$$y_{0j}^r + \sum_{\substack{i \in C \\ i \neq j}} y_{ij}^r = x_j^r \ \ j \in C,\ r \in L,\ r \neq N \tag{6}$$

$$y_{0j}^N = x_j^N \ \ j \in C \tag{7}$$

$$\sum_{i \in D} w_{ij}^r \geq y_{0j}^r \quad j \in C,\ r \in L \tag{8}$$

$$x_i^r \in \{0,1\} \quad i \in C,\ r \in L \tag{9}$$

$$y_{0j}^r \in \{0,1\} \quad j \in C,\ r \in L \tag{10}$$

$$y_{ij}^r \in \{0,1\} \quad i,\ j \in C,\ i \neq j,\ r \in L,\ r \neq N \tag{11}$$

$$w_{ij}^r \in \{0,1\} \quad i \in D,\ j \in C,\ r \in L \tag{12}$$

The objective function (1) minimizes the sum of arrival times to customers; here the coefficient r counts the number of times the travel time of a link is considered in the evaluation of the total latency. Constraints (2) guarantee that each customer is visited exactly once. Constraints (3) and (4) ensure the generation of exactly k routes. The constraints in (5)–(7) are the connectivity constraints and show the relation between the binary variables x_i^r and y_{ij}^r. Constraints in (5) require that any customer i visited at the upper level $r+1$ should be connected to exactly one customer (let say j) at level r by traversing edge (i, j). Constraints (6) impose that any customer j visited at level r should be linked to exactly one recently visited customer (let say i) by traversing edge (i, j) or linked directly to the auxiliary node by traversing edge $(0, j)$ in the same level. Constraints (7) require that customers visited at level N should be visited right after the node 0 by the link $(0, j)$ in the same level. Constraints (8) show the relation between binary variables w_{ij}^r and y_{0j}^r; if $y_{0j}^r = 1$, customer j is the first visited node at level r, hence it should follow the depot (let say i) in the same level r. Finally, constraints (9)–(12) establish the nature of the variables. This formulation has $n[N(m+n+1)-(n-1)]$ binary variables.

3.2 Second formulation

Another variant of the multi-depot layered network can be obtained by replacing each copy of node 0 in levels 2 to N, with m nodes, indexed from 1 to m, representing the multiple depots. In this way, levels from 2 to N include a copy of each customer and of each depot. In the first level, only the customer copies are present. Each tour in the network is represented by a route that ends in a first level customer and starts in a copy of the depot. Figure 2 illustrates this approach.

The way in which the layered network is defined implies that the linkage variables y_{0j}^r are replaced with binary variables y'_{ij^r} with $i in D$ and $j \in C$. Clearly, in this case, variables w_{ij}^r are no longer needed.
The second mathematical formulation is presented as follows.

$$\min z_2 = \sum_{i \in D} \sum_{j \in C} \sum_{r \in L} t_{ij} r y'_{ij^r} + \sum_{i \in C} \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{r \in L \\ r \neq N}} t_{ij} r y'_{ij^r} \tag{13}$$

$$\sum_{r \in L} x_{i^r} = 1 \quad i \in C \tag{14}$$

$$\sum_{r \in L} \sum_{i \in D} \sum_{j \in C} y'_{ij^r} = k \tag{15}$$

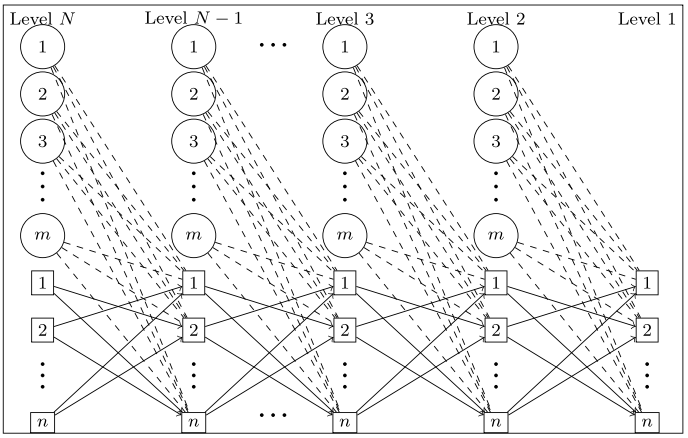

Fig. 2 Layered network for the MDk-TRP: second graphical representation

$$\sum_{i \in C} x_{i^1} = \sum_{r \in L} \sum_{i \in D} \sum_{j \in C} y'_{ij^r} \tag{16}$$

$$\sum_{\substack{j \in C \\ j \neq i}} y'_{ij^r} = x_{i^{r+1}} \quad i \in C,\ r \in L,\ r \neq N \tag{17}$$

$$\sum_{\substack{i \in V \\ i \neq j}} y'_{ij^r} = x_{j^r} \quad j \in C,\ r \in L,\ r \neq N \tag{18}$$

$$y'_{ij^N} = x_{i^N} \quad i \in D,\ j \in C \tag{19}$$

$$x_{i^r} \in \{0, 1\} \quad i \in C,\ r \in L \tag{20}$$

$$y'_{ij^r} \in \{0, 1\} \quad i \in D,\ j \in C,\ r \in L \tag{21}$$

$$y'_{ij^r} \in \{0, 1\} \quad i,\ j \in C,\ i \neq j,\ r \in L,\ r \neq N \tag{22}$$

Similar to (1), the objective function (13) is the waiting time of all customers and it is composed of two terms: the first one accounts for links originating from depots and the second one for the other links. In a similar way, the set of constraints (2)–(7) are equivalent to the set of constraints in (14)–(19) and the last set of constraints in (20)–(22) describe the nature of the variables. This new formulation reduces the number of variables to $n[(N(n+m) - (n-1)]$.

We should mention that for the special case of $|D| = 1$ (single depot), both models reduce to the classical k-TRP which is known to be NP-hard [16].

4 Metaheuristic approach

In this section, we describe a genetic algorithm to overcome the computational intractability of large size MDk-TRPs. In the following subsections, we first pro-

vide the outline of the metaheuristic approach an then detail the algorithm elements and the search mechanism.

4.1 Outline of the genetic algorithm

Genetic algorithms are evolutionary algorithms inspired by the theory of evolution, that use concepts such as natural selection, reproduction, genetic heritage and mutation [15]. Genetic algorithms may be enhanced with local search and/or other heuristic schemes. In this case, they are called hybrid genetic algorithms. Several routing problems have been successfully addressed by genetic algorithms [31,41,42].

Let P and $\hat{P}$ denote the old and the new population composed of p_s individuals (solutions/chromosomes); e_s represents the size of elite solutions, μ_r shows the mutation rate and p^* denotes the best solution with the fitness value $f(p^*)$. The pseudo-code of the proposed metaheuristic is presented in Algorithm 1. First, in Lines 2–6, the population set P is filled with randomly generated initial solutions $\{p_1, \ldots, p_i, \ldots, p_{p_s}\}$. In Line 7, the individuals are evaluated and sorted in non-decreasing order with respect to the fitness measure (see Sect. 4.4); also the best solution p^* is updated (Line 8).

A pre-specified portion $e_s \in [0.1, 1]$ of the fittest individuals, let say the elite solutions, are selected to directly go into the next population $\hat{P}$ (Lines 10–14). Then, a loop is run for $p_s - e_s$ iterations where at each iteration, a new individual of $\hat{P}$ is generated (Lines 15–30). Line 16 shows the tournament selection, a procedure which chooses a pair of parents from the population to be recombined using the crossover operator (Lines 17–18). The offsprings generated within the crossover phase are denoted with $\hat{p}_{(.)}$. In a local search procedure, the offsprings are educated (Line 19) and only the offspring with the lowest objective value ($\hat{p}$) is passed into the next generation (Lines 20–29). In Line 28, the mutation rate μ_r is updated appropriately (see *Mutation rate update* in Sect. 4.9). After, the new population set $\hat{P}$ is sorted (Line 31) and the sets P, $\hat{P}$ are updated. Lines 34–43 refer to the random restart mechanism which is, in fact, a diversification strategy that modifies the non-elite individuals in the population and updates the mutation rate, if necessary (Sect. 4.9). The algorithm ends at Line 45 and the best solution p^* is returned. Lines 9–44, are executed until the stopping criterion is met. In particular, we set a certain number of iterations proportional to the problem size.

4.2 Solution representation

Each solution in our algorithm is uniquely encoded by two ordered lists: one, called service pattern, with size n representing a giant tour, without trip delimiters. The order of the customers follows the visit order in the k routes. To specify the starting point of each route in the giant tour, we consider another list, called depot assignment, with size k, whose q^{th} element contains the index of the first customer visited in the q^{th} route and the index of the depot assigned to the route q. This TSP-like encoding style simplifies the implementation of crossover and mutation operators since it is easy to retrieve information from the solution [41].

Algorithm 1 The proposed metaheuristic algorithm

Initialization: $i,\ j \leftarrow 1,\ p_s,\ e_s,\ \mu_r,\ P,\ \hat{P} \leftarrow \emptyset,\ p^* \leftarrow 0,\ f(p^*) \leftarrow \infty$
while $(i \leq p_s)$ **do**
  $p_i \leftarrow Initial.Solution()$
  $P \leftarrow P \cup \{p_i\}$
  $i \leftarrow i + 1$
end while
$P \leftarrow fitness.sort(P)$
$p^* \leftarrow p_1,\ f(p^*) \leftarrow f(p_1)$
while stopping criterion is not met **do**
  $j \leftarrow 1$
  while $(|\hat{P}| < e_s)$ **do**
    $\hat{P} \leftarrow \hat{P} \cup \{p_j\}$
    $j \leftarrow j + 1$
  end while
  while $(|\hat{P}| < p_s)$ **do**
    $(p_1, p_2) \leftarrow Parent.Selection(P)$
    $\hat{p}_{(.)} \leftarrow Crossover(p_1, p_2)$
    $\hat{p}_{(.)} \leftarrow Mutation(\hat{p}_{(.)}, \mu_r)$
    $\hat{p}_{(.)} \leftarrow Local.Search(\hat{p}_{(.)})$
    if $f(\hat{p_1}) < f(\hat{p_2})$ **then**
      $\hat{p} \leftarrow \hat{p_1}$
    else
      $\hat{p} \leftarrow \hat{p_2}$
    end if
    $\hat{P} \leftarrow \hat{P} \cup \{\hat{p}\}$
    if $f(\hat{p}) < f(p^*)$ **then**
      $p^* \leftarrow \hat{p}$
      $Mutation.rate.Update(\mu_r)$
    end if
  end while
  $\hat{P} \leftarrow fitness.sort(\hat{P})$
  $P \leftarrow \hat{P}$
  $\hat{P} \leftarrow \emptyset$
  if (no improvement after a pre-specified number of iterations) **then**
    $Mutation.rate.Update(\mu_r)$
    for $(e_s < i \leq p_s)$ **do**
      $p_i \leftarrow random.solution()$
      if $f(p_i) < f(p^*)$ **then**
        $p^* \leftarrow \{p_i\}$
        $Mutation.rate.Update(\mu_r)$
      end if
    end for
  end if
end while
return p^*

Figure 3 displays the chromosome structure for a solution with three depots and ten customers:

$$d_3 - > c_5 - > c_8 - > c_9$$
$$d_1 - > c_2 - > c_4 - > c_7$$
$$d_1 - > c_3 - > c_1 - > c_6 - > c_{10}$$

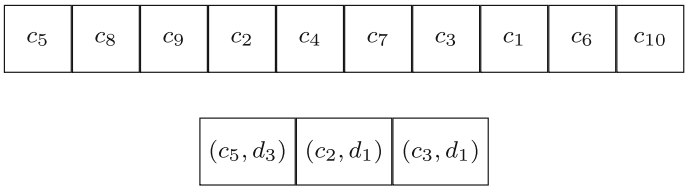

Fig. 3 Chromosome structure: a solution with three depots and ten customers

4.3 Initialization

The initialization procedure generates a pool of p_s feasible solutions and it is composed of two phases, including the *giant tour generation* and the *splitting heuristic*.

In the *giant tour generation* phase, an ordered sequence of customers is built in a greedy fashion. To do this, first, a non-visited customer is selected randomly. The giant tour is completed on the basis of the least-cost criterion until all the customers are inserted.

After, we apply a polynomial splitting heuristic proposed in [6,32] to find the optimal splitting points over the giant tour. This specifies the first customer to be visited in each route and also the optimal depot assigned to it. Obviously, once the starting point of a route is determined, we can retrieve it easily from the giant tour. The splitting can be done by solving a shortest-path problem on an auxiliary acyclic graph defined as $\mathcal{A} = (\mathcal{N}, \mathcal{D})$ in which $\mathcal{N} = \{0, 1, 2, \ldots, n\}$ and $\mathcal{D}$ is the set of possible subsequences of customers in the giant tour. The shortest path from 0 to n is calculated using Bellman's algorithm for directed acyclic graph (for more information, see [6,32]).

4.4 Fitness evaluation: *biased fitness* criterion

The performance of any population-based algorithm is affected by the *fitness measure* that enables to evaluate and order the individuals. We use the fitness criterion to select individuals for mating and to determine offsprings to keep in the next generation. In particular, we apply the *biased fitness function* criterion, $f(p_i)$ which is defined, for the individual p_i, as follows [41,42].

$$f(p_i) = fit(p_i) + \left(1 - \frac{e_s}{p_s}\right) \times dc(p_i) \tag{23}$$

where $fit(p_i)$ refers to the rank of p_i in the population, with respect to the objective value (total latency criterion) and $dc(p_i)$ is the rank of p_i with respect to the *diversity contribution* ($\Delta P(p_i)$) which is the average distance to its $\mathcal{N}_\mathcal{C}$ closest neighbours as given in (24).

$$\Delta P(p_i) = \frac{1}{|\mathcal{N}_\mathcal{C}|} \sum_{p_j \in \mathcal{N}_\mathcal{C}} \sigma^H(p_i, p_j) \tag{24}$$

$\sigma^H(p_i, p_j)$ is the normalized Hamming distance which quantifies the differences between two solutions p_i and p_j in terms of the visited customers and the depot assignment patterns. In general, $\mathcal{N}_C$ is set to $nc \times |P|$ where nc is an input parameter [41]. The inclusion of a diversity criterion into the fitness evaluation helps to avoid early convergence to local optimal solutions.

4.5 Parent selection

The parent selection procedure ($Parent.Selection(P)$) returns two different parents from the population set to generate new offsprings. The first parent is selected randomly (with uniform probability) from the top 50% solutions, evaluated and ranked based on the *biased fitness* criterion while the second parent is chosen randomly from the whole population.

4.6 Crossover operators

The genetic search progress is obtained by two essential genetic operations, including crossover (recombination) and mutation operators. As the name indicates, a crossover operator recombines the information of two parents based on some operator-specific rules and generates one or more offsprings (in our case, two offsprings), representing new solutions. Generally, the crossover operator exploits a better solution while the mutation operator explores a wider search space. In particular, we propose three crossover operators as follows:

- *One-point crossover*
  This operator picks a unique point randomly chosen in the range of $\phi \in (1, n)$, known as "crossover point" which divides each parent's giant tour into two segments with lengths of ϕ and $n - \phi$. The left segment of the first parent's giant tour and the right segment of the second parent's giant tour are directly copied into the first offspring. By changing the order of parents and with the same crossover point, the giant tour of the second offspring is built. The same process is repeated for the depot assignment lists of both parents where the cutting point, here, is picked from $(1, k)$, to be within the depot assignment list size.
- *Two-point crossover*
  This operator is similar to *One-point crossover* in which two random "crossover points" ϕ_1, ϕ_2 are selected to divide each parent's giant tour into three segments, including ϕ_1, $\phi_2 - \phi_1$, and $n - \phi_2$ customers. The middle segment of the first (second) parent's giant tour is copied exactly into the first (second) offspring as the middle part of its giant tour and the first and the third segments in the second (first) parent's giant tour are copied into the first and third segments of the offspring's customer service. In order to preserve the structure of the routes, the "crossover points" ϕ_1, ϕ_2 are selected in a biased way and pointing to the start of routes. The same procedure is also followed for depot assignment lists.
- *Uniform crossover*
  In the uniform crossover, each gene is selected randomly from one of the corre-

sponding genes of the parents with equal probability. The same rule is followed for building the depot assignment list.

4.7 Mutation

The mutation operator modifies the offsprings and acts as a diversification mechanism to avoid getting trapped into local optimal solutions. We propose four mutation operators that modify only the giant tours of the offsprings.

- *Intratour 3-swap*
  This operator randomly selects one route. If it visits more than three customers, a *3-swap* intratour operator is run swapping the position of three randomly selected customers. Otherwise, three random positions in the giant tour are selected and their contents are swapped. The latter case, in general, can be regarded as a *3-swap* intertour move.
- *Intertour 5-swap*
  This operator selects five randomly positions in the giant tour and swaps their contents.
- *Scramble*
  The *scramble* operator selects randomly a segment of the giant tour, and rearranges its content in a random fashion.
- *Inversion*
  Similar to the *Scramble*, the *Inversion* operator picks randomly a segment and reverses the order of the elements.

It is worth noting that after the mutation and the crossover, a repair mechanism is run to retain the feasibility of the solution, if needed. In general, the feasibility is checked with respect to set of visited customers and the number of dispatched vehicles.

If some of the customers have not been visited, they are inserted in the position leading to the least increase into the objective function. We should note that since the giant tour contains exactly n elements, in the latter case, there exist at least another customer visited more than once. Hence, whenever a non visited customer is inserted, one of the repeated customers is removed from the tour. Moreover, in the case that more than k vehicles have been dispatched, the extra tours with the highest latency are found and discarded.

4.8 Local search

The local search procedure explores the *2-swap* neighbourhood and updates the current solution whenever an improving move is detected (based on the first improving strategy). The neighbourhood is searched iteratively and the solution is updated until no further improvement is possible. The solution is modified to keep feasibility (from an open depot at least one vehicle should depart) and a right balancing among different tours (this measure is a proxy for near optimality of the solution). Each subtour is assigned to the best depot, based on the total latency criterion. Then, if it yields to an

improving solution, the tour lengths are rebalanced. A tour to can be splitted, if it is beneficial in terms of total latency, if the number of dispatched vehicles is less than k.

4.9 Diversification mechanism

In order to diversify the solution pool, the algorithm applies two different diversification strategies as given below. As we mentioned earlier, a diversity criterion is also included into the fitness evaluation avoiding the premature convergence of the population.

- *Random restart* In case the best solution (incumbent) is not updated within a pre-specified number of iterations, a strong diversification mechanism is applied. In particular, all the non-elite solutions in the population are replaced with new randomly generated ones.
- *Mutation rate update* The mutation probability is dynamically modified on the basis of the search history. Whenever this restart mechanism is executed, the mutation rate is increased to diversify the search.
  If the best solution is modified after the local search and the mutation rate value is greater than its initial value, the mutation rate is decreased to intensify the search.

5 Computational experiments

In this section, we report on the computational experiments carried out with the aim of testing the proposed models as well as the solution approach. We first assessed the efficiency of the two mathematical formulations, implemented using the AMPL optimization language [10] and solved by Gurobi 8.1.0 [14] within a time limit of 7200 s. To have a comparison, we also re-implemented (excluding the constraints on the capacity of vehicles) the models proposed for the multi-depot CCVRP, namely the one proposed in [19] and [44]. We refer to these models by the authors' names and denote our first and second mathematical formulations in Sect. 3 by F1 and F2, respectively. As test bed, we considered three sets of instances taken from the literature [7,19][1]. The algorithm was coded in C++ language and all the experiments were executed on an Intel Core i5, with 2.4 GHz CPU and 8 GB RAM.

5.1 Results obtained considering benchmark instances

Tables 1 and 2 report the computational results obtained by Gurobi, in solving the mathematical models within the time limit. For each considered model, we report the objective value of the best solution found (OF), the computational time, in seconds, (CPU), and the optimality gap (Gap), in percentage, reported by Gurobi. The Tables also report when time limit TL is reached and a dash for instances in which no feasible solution within the time limit was found. The instance name is reported in the column with the header "Instance".

[1] http://www.bernabe.dorronsoro.es/vrp/; https://github.com/elalla/MDCCVRP.

Table 1 Modeling comparison for the set of lr instances

Instance				Lalla's model			Wang's model			F1			F2		
Id	n	m	k	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap
lr01	10	4	5	545.69	4.02	0	545.69	5.53	0	545.69	0.16	0	545.69	1.02	0
lr02	10	4	5	832.69	11.32	0	832.69	6.02	0	832.69	0.15	0	832.69	0.15	0
lr03	10	4	5	832.78	13.53	0	832.78	9.84	0	832.78	0.16	0	832.78	0.47	0
lr04	25	4	10	2082.27	TL	33.62	2178.08	TL	44.22	2082.28	0.16	0	2082.28	0.22	0
lr05	25	4	10	1836.99	TL	36.28	1943.43	TL	43.44	1827.41	1.13	0	1827.41	0.22	0
lr06	25	4	10	1788.11	TL	29.97	1786.75	TL	38.51	1786.95	0.13	0	1786.95	0.22	0
lr07	50	4	20	6688.96	TL	81.60	6349.98	TL	99.43	5424.57	2.73	0	5424.57	3.13	0
lr08	50	4	20	5090.32	TL	86.29	5119.09	TL	99.29	3737.38	2.89	0	3737.38	4.14	0
lr09	50	4	20	4638.48	TL	82.52	4610.18	TL	99.31	3802.88	2.08	0	3802.88	4.13	0
lr10	50	6	25	2830.97	TL	54.13	9196.97	TL	99.97	2786.43	1.28	0	2786.43	2.13	0
lr11	50	6	25	2928.25	TL	53.24	3008.81	TL	98.98	2909.27	2.52	0	2909.27	3.44	0
lr12	50	6	25	4498.86	TL	85.38	4135.15	TL	99.13	3171.75	2.42	0	3171.75	3.27	0
lr13	100	4	25	–	TL	–	–	TL	–	8288.43	51.80	0	8288.43	61.95	0
lr14	100	4	25	–	TL	–	–	TL	–	7257.31	39.73	0	7257.31	50.05	0
lr15	100	4	25	–	TL	–	–	TL	–	8625.13	32.74	0	8625.13	43.94	0
lr16	100	6	25	–	TL	–	–	TL	–	5265.30	57.80	0	5265.30	47.33	0
lr17	100	6	25	38,994.49	TL	99.23	–	TL	–	6107.32	35.77	0	6107.32	46.97	0
lr18	100	6	25	84,725.04	TL	99.70	–	TL	–	5788.73	76.78	0	5788.73	49.95	0
Avg.					6001.60	53.00		6001.19	60.19	3948.46	17.25	0		17.93	0

Table 2 Modeling comparison for the sets p and pr instances ($k = 35$)

Instance			Lalla's model			Wang's model			F1			F2		
Id	n	m	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap
p01	50	4	685.30	TL	48.19	684.52	TL	100	660.34	1.05	0	660.34	4.13	0
p02	50	4	685.30	TL	48.19	684.52	TL	100	660.34	1.08	0	660.34	5.72	0
p03	75	5	1596.34	TL	92.96	1303.76	TL	100	906.24	10.23	0	906.24	20.91	0
p04	100	2	4551.33	TL	94.17	4730.70	TL	100	1881.48	55.75	0	1881.48	57.94	0
p05	100	2	37,953.96	TL	99.30	6116.31	TL	100	1871.62	54.64	0	1871.62	59.92	0
p06	100	3	4655.80	TL	96.32	7444.78	TL	100	1460.60	49.66	0	1460.60	54.91	0
p07	100	4	4608.38	TL	97.18	–	TL	–	1453.64	43.94	0	1453.64	80.95	0
p08	249	2	–	TL	–	–	TL	–	–	TL	–	–	TL	–
p09	249	3	–	TL	–	–	TL	–	12,464.53	TL	1.76	–	TL	–
p10	249	4	–	TL	–	–	TL	–	11,782.90	TL	4.55	12971.50	TL	13.80
p11	249	5	–	TL	–	–	TL	–	11,529.61	TL	3.27	–	TL	–
p12	80	2	4146.07	TL	85.98	7357.42	TL	100	2769.07	26.30	0	2769.07	28.48	0
p15	160	4	–	TL	–	–	TL	–	5618.84	1062.89	0	5618.84	1988.09	0
p18	240	6	–	TL	–	–	TL	–	11,311.64	TL	0.70	11311.64	TL	1.55

Table 2 continued

Instance			Lalla's model			Wang's model			F1			F2		
Id	n	m	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap
pr01	48	4	1192.87	TL	46.07	1211.99	TL	100	1167.74	1.23	0	1167.74	6.31	0
pr02	96	4	–	TL	–	–	TL	–	2422.94	50.58	0	2422.94	118.06	0
pr03	144	4	–	TL	–	–	TL	–	4287.30	870.06	0	4287.30	492.36	0
pr04	192	4	–	TL	–	–	TL	–	5582.29	6061.16	0	5582.29	2992.41	0
pr05	240	4	–	TL	–	–	TL	–	6782.96	3647.80	0	6782.96	7057.45	0
pr06	288	4	–	TL	–	–	TL	–	–	TL	–	–	TL	–
pr07	72	6	2219.60	TL	79.96	4879.88	TL	100	1594.15	7.20	0	1594.15	15.36	0
pr08	144	6	–	TL	–	–	TL	–	3817.70	758.08	0	3817.70	352.63	0
pr09	216	6	–	TL	–	–	TL	–	5668.45	2484.63	0	5668.45	2698.86	0
pr10	288	6	–	TL	–	–	TL	–	–	TL	–	–	TL	–
Avg.				TL	78.83		TL	100		2732.76	0.49		2768.10	0.81

From the results in Table 1, we can observe that F1 and F2 found the optimal solutions for all the instances. The models adapted from [19] and [44] were able to find the optimal solution for the smallest instances with 10 customers. For the instances with 25 customers, after two hours, we observed considerably optimality gaps of at least 29% (Lalla's model) and 38% (Wang's model). For all the instances with more than 50 customers, the models provided feasible solutions with optimality gap of about 53% and 99%, respectively. we should remark that, in this respect, the two models were proposed for the multi-depot CCVRP. Nevertheless, our models could be used to set good lower bound in reasonable computational time. In fact, in terms of solution time, F1 and F2 are significantly faster than the other models, and the largest Ir instance including 100 customers is solved to optimality in about 76 s.
Table 2 summarizes the results for the set of p and pr instances, including up to 288 customers with a fleet size of $k = 35$ [7]. As shown in Table 2, the results of the proposed models F1 and F2 are superior to those obtained from other existing models. In particular, model F1 finds optimal solutions in at least 70% of cases, only in three cases no feasible solution is found and for those cases not solved to optimality, the gap is always less than 5%.

In addition, while, compared to F2, model F1 involves more binary variables, its computational time is significantly lower for all the instances solved to optimality. This behaviour is not related to the tightness of its LP relaxation bound. Tables 3 and 4 report the values of the linear relaxations of F1 and F2 and the gaps with respect to the Best Known Solution (BKS), in percentage. Both F1 and F2 present similar LP relaxation bounds, except for instances p08, p10, p18, pr01 and pr05 where the difference is less than 0.01%.

In the following, we present the results of the genetic algorithm. Specifically, we considered a population size of 25 individuals ($p_s = 25$); the initial mutation rate was set to $\frac{1}{6}$ and its possible values lie in the set $\mu_r = \frac{1}{\gamma}$, $\gamma \in \{1, 2, \ldots, 6\}$. The elitism rate e_s was set to $\lfloor 40\% \times p_s \rfloor$ and the parameter nc was set to 0.2. The maximum number of iterations was set to $\frac{10^4 \times f}{p_s}$ where, depending on the instance, $f \in \{1, 2, 5, 10, 15, 20\}$. Finally, the random restart mechanism is activated every 100 iterations without incumbent update. The genetic algorithm was tested considering two different configurations: with one-point crossover and scramble operators used for reproduction and mutation, respectively and with operators chosen randomly (with uniform probability) among the set of operators defined in Sects. 4.6 and 4.7.

Table 5 summarizes the metaheuristic results. Columns 1–4 show the instance features; Columns 5 and 6 represent the objective value of the BKS (obtained from model F1 or F2), and its corresponding solution time (he symbol "–" in the Column with header OF denotes that the solver could not obtain any feasible solution within the time limit). Columns 7 and 8 report the splitting algorithm results, including the best objective value (OF) and the relative gap, in percentage, (Gap) calculated with respect to the Gurobi OF, if available; otherwise denoted by "–". Columns 9–12 and 13–16, respectively, display the results of the genetic algorithm with and without the random selection of the operators. The column with heading ΔT reports the percentage time fraction, calculated as ($\Delta T = \frac{CPU_{heu}}{CPU_{Gurobi}} \times 100$) where CPU_{heu} and CPU_{Gurobi} denote the CPU of the metaheuristic and Gurobi, respectively.

Table 3 Linear relaxations for the set of lr instances

Instance	BKS	F1 relaxation		F2 relaxation	
		OF	$Gap(\%)$	OF	$Gap(\%)$
lr01	545.69	545.69	0	545.69	0
lr02	832.69	832.69	0	832.69	0
lr03	832.78	830.81	0.24	830.81	0.24
lr04	2082.28	2082.28	0	2082.28	0
lr05	1827.41	1827.41	0	1827.41	0
lr06	1786.95	1786.95	0	1786.95	0
lr07	5424.57	5424.57	0	5424.57	0
lr08	3737.38	3733.37	0.11	3733.37	0.11
lr09	3802.88	3796.40	0.17	3796.40	0.17
lr10	2786	2786	0	2786	0
lr11	2909.27	2909.27	0	2909.27	0
lr12	3171.75	3166.19	0.18	3166.19	0.18
lr13	8293.42	8287.28	0.07	8287.28	0.07
lr14	7273.03	7253.90	0.26	7253.90	0.26
lr15	8626.13	8625.13	0.01	8625.13	0.01
lr16	5306.50	5251.50	1.03	5251.50	1.03
lr17	6141.07	6098.79	0.69	6098.79	0.69
lr18	5804.19	5781.86	0.39	5781.86	0.39
Avg.			0.18		0.18

The results show that the average solution time of the heuristic over all the instances is less than 3 min, compared to Gurobi that in 12% of cases could not find any feasible solution within two hours. The effectiveness of the method is also confirmed by the average ΔT values reported in Columns 12 and 16 which are around 15% and 22%, respectively. This means that the metaheuristic is about 7 and 5 times faster than Gurobi.

In addition, the metaheuristic finds optimal solutions for instances up to 25 customers. Even the splitting algorithm performs well since average solution gap is below 12.37% and in 3 cases (for which the Gurobi could not find any feasible solution within two hours), the splitting algorithm finds good feasible solutions. To have a clear idea about the performance of the metaheuristic compared with the splitting algorithm, we focus on the most challenging cases, for which Gurobi did not find any feasible solution. For such cases, the metaheuristics provide solutions which are on average 9.60% (with a minimum of 5.80% and a maximum of 19.74%) better than the splitting algorithm solution.

The results show a slightly better performance (about 0.20%), in terms of percentage gaps, when the random selection is present (see Columns 11 and 15). Although, in this case, the average computational time is lower, the speed up ratio ΔT deteriorates.

In summary, for small instances optimal solutions are obtained in a negligible solution time, for medium instances good quality solutions can be obtained within

Table 4 Linear relaxations for the set of p and pr instances

Instance	BKS	F1 relaxation		F2 relaxation	
		OF	$Gap(\%)$	OF	$Gap(\%)$
p01	660.34	652.78	1.14	652.78	1.14
p02	660.34	652.78	1.14	652.78	1.14
p03	906.24	899.68	0.72	899.68	0.72
p04	1881.48	1855.76	1.37	1855.76	1.37
p05	1871.62	1865.33	0.34	1865.33	0.34
p06	1460.60	1449.50	0.76	1449.50	0.76
p07	1453.64	1446.47	0.49	1446.47	0.49
p08	–	13,779.30	–	13,779.27	–
p09	12,464.53	11,905.67	4.48	11,905.67	4.48
p10	11,782.90	10,980.44	6.81	10,980.40	6.81
p11	11,529.61	10,875.72	5.67	10,875.72	5.67
p12	2769.07	2717.44	1.86	2717.44	1.86
p15	5618.84	5589.14	0.53	5589.14	0.53
p18	11,311.64	10,826.80	4.28	10,826.76	4.28
pr01	1167.74	1135.24	2.78	1133.4	2.78
pr02	2422.94	2347.78	3.10	2347.78	3.10
pr03	4287.30	4132.45	3.61	4132.45	3.61
pr04	5582.29	5465.36	2.11	5465.36	2.11
pr05	6782.96	6506.25	4.08	6506.24	4.08
pr06	–	8003.89	–	8003.89	–
pr07	1594.15	1568.62	1.60	1568.62	1.60
pr08	3817.70	3722.63	2.49	3722.63	2.49
pr09	5668.45	5452.80	3.87	5452.80	3.87
pr10	–	8304.91	–	8304.91	–
Avg.			2.53		2.53

acceptable computational time, and for the largest and the most challenging cases, for which Gurobi does not report any feasible solution, the algorithm is able to find solutions with gaps, calculated with respect to the lower bound, less than 23%,on average.

5.2 Results considering a reduced fleet size

With the aim of assessing the impact of the fleet size on the model and on the heuristic performance, we carried out an additional set of experiments, on the p, pr, and lr instances, considering a different number of vehicles. As a matter of fact, we observed that the optimal routes obtained considering the original datasets, contain only a few vehicles, maybe because they were originally proposed for the multi-

Table 5 The metaheuristic results

Instance				BKS		Splitting heuristic		Genetic algorithm				Genetic algorithm with random selection			
Id	n	m	k	OF	CPU	OF	Gap	OF	CPU	Gap	ΔT	OF	CPU	Gap	ΔT
lr01	10	4	5	545.69	0.16	545.69	0	545.69	0	0	0	545.69	0	0	0
lr02	10	4	5	832.69	0.15	832.69	0	832.69	0	0	0	832.69	0	0	0
lr03	10	4	5	832.78	0.16	832.78	0	832.78	0	0	0	832.78	0	0	0
lr04	25	4	10	2082.28	0.16	2082.28	0	2082.28	0	0	0	2082.28	0	0	0
lr05	25	4	10	1827.41	0.22	1827.41	0	1827.41	0	0	0	1827.41	0	0	0
lr06	25	4	10	1786.95	0.13	1786.95	0	1786.95	0	0	0	1786.95	0	0	0
lr07	50	4	20	5424.57	2.73	5515.93	1.68	5446.16	0	0.40	0	5444.57	0	0.37	0
lr08	50	4	20	3737.38	2.89	3872.91	3.63	3761.86	0	0.66	0	3760.03	0	0.61	0
lr09	50	4	20	3802.88	2.08	3882.58	2.10	3834.01	0	0.82	0	3812.65	0	0.26	0
lr10	50	6	25	2786.43	1.28	3072.37	10.26	3022.37	0	8.47	0	2986.69	0	7.19	0
lr11	50	6	25	2909.27	2.52	3163.43	8.74	3117.34	0	7.15	0	3107.61	0	6.82	0
lr12	50	6	25	3171.75	2.42	3252.61	2.55	3204.09	0	1.02	0	3186.67	0	0.47	0
lr13	100	4	25	8288.43	51.80	8904.55	7.43	8563.04	13.07	3.31	25.23	8593.57	22.01	3.68	42.49
lr14	100	4	25	7257.31	39.73	7832.69	7.93	7519.98	16.84	3.62	42.39	7538.32	21.98	3.87	55.32
lr15	100	4	25	8625.13	32.74	9084.78	5.33	8840.64	16.11	2.50	49.21	8803.73	21.97	2.07	67.10
lr16	100	6	25	5265.30	47.33	5901.06	12.07	5425.24	21.28	3.04	44.96	5399.68	21.96	2.55	46.40
lr17	100	6	25	6107.32	35.77	6754.79	10.60	6353.43	17.81	4.03	49.79	6362.25	21.96	4.17	61.39
lr18	100	6	25	5788.73	49.95	6408.12	10.70	6017.02	12.44	3.94	24.90	6025.97	21.97	4.10	43.98
p01	50	4	35	660.34	1.05	959.61	45.32	959.17	0	45.25	0	959.12	0	45.25	0
p02	50	4	35	660.34	1.08	713.47	8.05	712.49	0	7.90	0	712.49	0	7.90	0
p03	75	5	35	906.24	10.23	980.67	8.21	963.97	4.55	6.37	44.48	958.69	6.73	5.79	65.79

Table 5 continued

Instance				BKS		Splitting heuristic		Genetic algorithm				Genetic algorithm with random selection			
Id	n	m	k	OF	CPU	OF	Gap	OF	CPU	Gap	ΔT	OF	CPU	Gap	ΔT
p04	100	2	35	1881.48	55.75	1995.38	6.05	1970.90	28.97	4.75	51.96	1976.45	21.97	5.05	39.41
p05	100	2	35	1871.62	54.64	2028.59	8.39	2006.38	4.63	7.20	8.47	2002.56	22.03	7.00	40.32
p06	100	3	35	1460.60	49.66	1626.06	11.33	1587.70	24.73	8.70	49.80	1598.06	21.98	9.41	44.26
p07	100	4	35	1453.64	43.94	1603.73	10.33	1546.16	26.63	6.36	60.61	1548.89	21.96	6.55	49.98
p08	249	2	35	–	TL	17,298.57	–	16,220.42	166.28	–	2.31	16,281.50	274.53	–	3.81
p09	249	3	35	12,464.53	TL	15,354.70	23.19	14,226.97	479.22	14.14	6.66	14,129.02	274.71	13.35	3.82
p10	249	4	35	11,782.90	TL	14,194.45	20.47	13,309.76	426.69	12.96	5.93	13,087.11	274.52	11.07	3.81
p11	249	5	35	11,529.61	TL	14,444.53	25.28	13,344.04	335.58	15.74	4.66	13,282.14	274.66	15.20	3.81
p12	80	2	35	2769.07	26.30	2915.82	5.30	2897.06	1.91	4.62	7.26	2897.06	6.72	4.62	25.55
p15	160	4	35	5618.84	1062.89	9254.87	64.71	8609.33	312.97	53.22	29.45	8563.86	219.85	52.41	20.68
p18	240	6	35	11,311.64	TL	15,427.88	36.39	12,397.36	654.49	9.60	9.09	12,033.14	366.14	6.38	5.09
pr01	48	4	35	1167.74	1.23	1247.05	6.79	1243.84	0	6.52	0	1244.15	0	6.54	0
pr02	96	4	35	2422.94	50.58	2652.49	9.47	2576.27	5.56	6.33	10.99	2589.48	54.92	6.87	108.58
pr03	144	4	35	4287.30	492.36	4914.50	14.63	4715.83	21.47	10	4.36	4636.24	109.87	8.14	22.31
pr04	192	4	35	5582.29	2992.41	6567.38	17.65	6073.17	289.46	8.79	9.67	6026.25	219.70	7.95	7.34
pr05	240	4	35	6782.96	3647.80	8722.00	28.59	7527.32	210.87	10.97	5.78	7609.85	274.53	12.19	7.53
pr06	288	4	35	–	TL	11,440.42	–	9554.50	296.96	–	4.12	9492.53	368.36	–	5.12
pr07	72	6	35	1594.15	7.20	1752.03	9.90	1723.57	1.82	8.12	25.28	1726.42	6.73	8.30	93.47
pr08	144	6	35	3817.70	352.63	4350.86	13.97	4077.16	129.42	6.80	36.70	4093.12	109.83	7.21	31.15
pr09	216	6	35	5668.45	2484.63	7113.50	25.49	6291.82	305.36	11.00	12.29	6404.55	183.05	12.99	7.37
pr10	288	6	35	–	TL	18,599.76	–	17,574.79	289.28	–	4.02	17,667.65	366.14	–	5.09
Avg.					1476.35		12.37		97.96	7.80	15.01		85.97	7.60	21.69

Table 6 Modeling comparison: set of lr, p, and pr instances with reduced fleet size

Instance				Lalla's model			Wang's model			F1			F2		
Id	n	m	k	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap
lr01	10	4	4	592.27	6.15	0	**592.27**	4.50	0	**592.27**	0.05	0	**592.27**	0.08	0
lr02	10	4	4	**885.89**	10.44	0	**885.89**	11.58	0	**885.89**	0.06	0	**885.89**	0.08	0
lr03	10	4	4	**846.91**	7.62	0	**846.91**	9.55	0	**846.91**	0.06	0	**846.91**	0.09	0
lr04	25	4	4	**3267.63**	TL	97.20	**3267.63**	TL	64.49	**3267.63**	5.59	0	**3267.63**	4.77	0
lr05	25	4	4	3383.12	TL	96.90	3321.94	TL	65.34	**3267.66**	5.86	0	**3267.66**	4.78	0
lr06	25	4	4	3362.70	TL	96.60	**2965.13**	TL	61.62	**2965.13**	4.97	0	**2965.13**	5.44	0
lr07	50	4	5	11159.96	TL	98.30	10972.3	TL	99.67	**8325.97**	225.06	0	**8325.97**	544.92	0
lr08	50	4	5	8387.22	TL	98.70	7586.68	TL	99.42	**7089.64**	2258.77	0	**7089.64**	6719.86	0
lr09	50	4	5	8541.46	TL	98.70	8557.22	TL	99.71	**7390.97**	291.95	0	**7390.97**	1200.64	0
lr10	50	6	6	6824.19	TL	98.40	6661.41	TL	99.33	**6143.47**	631.83	0	**6143.47**	700.11	0
lr11	50	6	6	6245.32	TL	98.00	–	TL	–	**5756.80**	585.50	0	**5756.80**	352.25	0
lr12	50	6	6	6552.02	TL	98.02	6985.64	TL	99.36	**5699.16**	141.33	0	**5699.16**	139.88	0
lr13	100	4	10	–	TL	–	–	TL	–	**11,744.60**	4394.58	0	**11,744.60**	5375.69	0
lr14	100	4	10	–	TL	–	–	TL	–	**10,742.60**	733.94	0	**10,742.60**	1073.09	0
lr15	100	4	10	–	TL	–	–	TL	–	**11,744.60**	4243.97	0	**11,744.60**	5006.88	0
lr16	100	6	10	–	TL	–	–	TL	–	**9442.62**	2971.83	0	**9442.62**	5799.20	0
lr17	100	6	10	–	TL	–	–	TL	–	**9917.56**	1647.61	0	**9917.56**	2136.41	0
lr18	100	6	10	–	TL	–	–	TL	–	**9220.77**	2907.75	0	**9220.77**	5040.42	0
p01	50	4	5	2141.17	TL	98.87	2182.70	TL	99.57	**1958.95**	228.547	0	**1958.95**	127.70	0
p02	50	4	5	3954.09	TL	99.39	2297.82	TL	99.59	**1958.95**	220.453	0	**1958.95**	127.00	0
p03	75	5	8	3386.14	TL	99.03	–	TL	–	**2271.22**	5458.62	0	**2271.22**	3091.94	0

Table 6 continued

Instance				Lalla's model			Wang's model			F1			F2		
Id	n	m	k	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap	OF	CPU	Gap
p04	100	2	10	8456.15	TL	99.25	7071.87	TL	99.85	**3122.13**	2020.05	0	**3122.13**	1595.27	0
p05	100	2	10	10074.8	TL	99.49	–	TL	–	**3103.26**	893.641	0	**3103.26**	568.53	0
p06	100	3	10	–	TL	–	6173.60	TL	99.84	**2870.05**	1141.33	0	**2870.05**	821.25	0
p07	100	4	10	–	TL	–	4970.64	TL	99.87	**2899.07**	6147.14	0	**2899.07**	6284.33	0
p08	249	2	25	–	TL	–	–	TL	–	–	TL	–	**16,620.90**	5155.70	0
p09	249	3	25	–	TL	–	–	TL	–	14833.00	TL	1.66	**14,809.50**	TL	1.65
p10	249	4	25	–	TL	–	–	TL	–	–	TL	–	**14,116.10**	TL	3.43
p11	249	5	25	–	TL	–	–	TL	–	–	TL	–	**15,189.20**	TL	10.92
p12	80	2	8	9497.25	TL	99.04	7334.81	TL	99.81	**5479.50**	5514.27	0	**5479.50**	5584.90	0
p15	160	4	16	–	TL	–	–	TL	–	**10,370.70**	TL	4.47	**10,370.70**	TL	4.47
p18	240	6	24	–	TL	–	–	TL	–	**15,682.10**	TL	3.71	15,863.80	TL	3.75
pr01	48	4	5	3329.63	TL	99.86	3320.66	TL	99.50	**3036.43**	234.484	0	**3036.43**	269.97	0
pr02	96	4	10	15632.22	TL	99.60	7109.70	TL	99.73	**4092.51**	2015.23	0	**4092.51**	1820.75	0
pr03	144	4	15	–	TL	–	–	TL	–	**6474.18**	3893.82	0	**6474.18**	7105.80	0
pr04	192	4	20	–	TL	–	–	TL	–	**7102.26**	4364.53	0	**7102.26**	6970.50	0
pr05	240	4	24	–	TL	–	–	TL	–	9338.87	TL	14.02	**8157.22**	TL	12.20
pr06	288	4	29	–	TL	–	–	TL	–	9404.57	TL	2.56	**9366.42**	TL	2.55
pr07	72	6	8	4272.66	TL	98.52	6636.91	TL	99.77	**3496.68**	248.906	0	**3496.68**	404.06	0
pr08	144	6	15	–	TL	–	–	TL	–	**5906.88**	6485.65	0	**5906.88**	6057.30	0
pr09	216	6	22	–	TL	–	–	TL	–	7315.38	TL	0.74	**7309.01**	2826.55	0
pr10	288	6	29	–	TL	–	–	TL	–	10,669.96	TL	9.45	**9869.74**	TL	1.17
Avg.					6686.29	84.47		6686.32	79.32		3140.89	0.94		3345.62	0.96

Table 7 The metaheuristic results: instances with reduced fleet size

Instance				BKS		Splitting heuristic		Genetic algorithm				Genetic algorithm with random selection			
Id	n	m	k	OF	CPU	OF	Gap	OF	CPU	Gap	ΔT	OF	CPU	Gap	ΔT
lr01	10	4	4	592.27	0.05	592.27	0	592.27	0	0	0	592.27	0	0	0
lr02	10	4	4	885.89	0.06	885.89	0	885.89	0	0	0	885.89	0	0	0
lr03	10	4	4	846.91	0.06	846.91	0	846.91	0	0	0	846.91	0	0	0
lr04	25	4	4	3267.63	4.77	3267.63	0	3267.63	0	0	0	3267.63	0	0	0
lr05	25	4	4	3267.66	4.78	3267.66	0	3267.66	0	0	0	3267.66	0	0	0
lr06	25	4	4	2965.13	4.97	2965.13	0	2965.14	0	0	0	2965.14	0	0	0
lr07	50	4	5	8325.97	225.06	8475.87	1.80	8371.32	4.30	0.54	1.91	8385.10	2.02	0.71	0.90
lr08	50	4	5	7089.64	2258.77	7348.94	3.66	7143.59	19.33	0.76	0.86	7160.89	15.93	1.00	0.71
lr09	50	4	5	7390.97	291.95	7566.80	2.38	7468.66	1.44	1.05	0.49	7417.23	2.39	0.36	0.82
lr10	50	6	6	6143.47	631.83	6775.57	10.29	6669.28	0.22	8.56	0.03	6598.34	0.34	7.40	0.05
lr11	50	6	6	5756.80	352.25	6265.09	8.83	6168.67	6.83	7.15	1.94	6171.19	0.94	7.20	0.27
lr12	50	6	6	5699.16	139.88	5865.22	2.91	5776.82	2.16	1.36	1.54	5730.72	0.88	0.55	0.63
lr13	100	4	10	11,744.60	4394.58	12,646.98	7.68	12,161.63	983.64	3.55	22.38	12,191.38	1579.41	3.80	35.94
lr14	100	4	10	10,742.60	733.94	11,601.06	7.99	11,160.95	255.52	3.89	34.81	11,188.98	327.13	4.16	44.57
lr15	100	4	10	11,744.60	4243.97	12,415.45	5.71	12,059.63	1575.57	2.68	37.12	12,033.57	2131.79	2.46	50.23
lr16	100	6	10	9442.62	2971.83	10,598.56	12.24	9763.21	1337.39	3.40	45	9685.37	1379.60	2.57	46.42
lr17	100	6	10	9917.56	1647.61	11,007.69	10.99	10,343.79	637.77	4.30	38.71	10,349.80	771.92	4.36	46.85
lr18	100	6	10	9220.77	2907.75	10,240.80	11.06	9604.71	759.75	4.16	26.13	9610.15	1291.55	4.22	44.42
p01	50	4	5	1958.95	127.70	2851.48	45.56	2853.17	0.03	45.65	0.02	2852.25	1.53	45.60	1.20
p02	50	4	5	1958.95	127.00	2117.00	8.07	2117.80	0.37	8.11	0.29	2118.76	1.54	8.16	1.21
p03	75	5	8	2271.22	3091.94	2461.18	8.36	2416.13	1249.57	6.38	40.41	2407.24	1806.63	5.99	58.43

Table 7 continued

Instance				BKS		Splitting heuristic		Genetic algorithm				Genetic algorithm with random selection			
Id	n	m	k	OF	CPU	OF	Gap	OF	CPU	Gap	ΔT	OF	CPU	Gap	ΔT
p04	100	2	10	3122.13	1595.27	3317.80	6.27	3271.47	1018.95	4.78	63.87	3285.54	769.38	5.23	48.23
p05	100	2	10	3103.26	568.53	3368.91	8.56	3333.14	80.36	7.41	14.13	3320.63	332.87	7.00	58.55
p06	100	3	10	2870.05	821.25	3205.03	11.67	3126.83	530.66	8.95	64.62	3148.09	457.47	9.69	55.70
p07	100	4	10	2899.07	6147.14	3199.61	10.37	3087.86	2130.18	6.51	34.65	3097.86	1669.91	6.86	27.17
p08	249	2	25	16,620.90	5155.70	16,664.05	0.26	16,648.23	141.19	0.16	2.74	16,652.11	229.42	0.19	4.45
p09	249	3	25	14,809.50	TL	14,858.20	0.33	14,866.31	542.10	0.38	7.53	14,816.49	281.66	0.05	3.91
p10	249	4	25	14,116.10	TL	15,451.26	9.46	14,487.06	561.07	2.63	7.79	14,255.40	321.24	0.99	4.46
p11	249	5	25	15,189.20	TL	15,228.23	0.26	15,227.14	450.13	0.25	6.25	15,209.67	294.63	0.13	4.09
p12	80	2	8	5479.50	5514.27	5776.69	5.42	5739.15	443.16	4.74	8.04	5748	1360.34	4.90	24.67
p15	160	4	16	10,370.70	TL	17,110.50	64.99	15,891.50	1226.08	53.23	17.03	15,834.43	857.73	52.68	11.91
p18	240	6	24	15,682.10	TL	21,688.93	38.30	17,389.48	792.08	10.89	11.00	16,884.47	381.01	7.67	5.29
pr01	48	4	5	3036.43	234.48	3245.08	6.87	3245.12	4.47	6.87	1.91	3245.92	1.09	6.90	0.46
pr02	96	4	10	4092.51	1820.75	4490.35	9.72	4367.49	120.23	6.72	6.60	4380.66	857.58	7.04	47.10
pr03	144	4	15	6474.18	3893.82	7441.05	14.93	7136.31	341.05	10.23	8.76	7024.86	1594.11	8.51	40.94
pr04	192	4	20	7102.26	4364.53	8363.73	17.76	7750.59	693.93	9.13	15.90	7687.69	528.79	8.24	12.12
pr05	240	4	24	8157.22	TL	10,507.46	28.81	9061.75	347.30	11.09	4.82	9166.04	303.71	12.37	4.22
pr06	288	4	29	9366.42	TL	9368.55	0.02	9392.40	346.70	0.28	4.82	9385.54	374.06	0.20	5.20
pr07	72	6	8	3496.68	248.91	3855.07	10.25	3781.00	49.79	8.13	20	3798.73	180.13	8.64	72.37
pr08	144	6	15	5906.88	6057.30	6734.45	14.01	6319.81	2251.73	6.99	37.17	6335.11	1929.76	7.25	31.86
pr09	216	6	22	7309.01	2826.55	9196.37	25.82	8128.12	372.10	11.21	13.16	8262.81	204.69	13.05	7.24
pr10	288	6	29	9869.74	TL	9888.27	0.19	9876.42	313.43	0.07	4.35	9876.53	384.47	0.07	5.34
Avg.					2881.17		10.28		466.44	6.48	14.45		538.75	6.34	19.24

depot CCVRP. Therefore, we have considered a different number of vehicles, setting $k = max\{m, \lceil \frac{n}{10} \rceil\}$, to make our problem more challenging.

Table 6 shows the results; for the sake of completeness, we also report the solutions of both Lalla's and Wang's model imposing a time limit of two hours. The best objective function value is highlighted in bold. As confirmed by the gap values reported in Columns 7 and 10, the extant models could only solve to optimality the smallest Ir instances with 10 customers. For moderate instances, with 25 customers, the optimality gaps after two hours are very high (at least 61.62% for the Wang's model and 96.60% for the Lalla's one). When the number of customers is increased to 100, none of the existing models could find any feasible solution within the time limit. Nevertheless, both F1 and F2 solved all the set of lr instances to optimality, even if the decrease in the fleet size considerably increases the solution time. We also observe that the second formulation provides slightly better solutions, but in terms of solution time, model F1 is superior to F2.

Regarding the p and pr instances, we noticed that the existing models fail to find optimal solutions. In particular, the Lalla's model obtained feasible solutions for instances up to 100 nodes and two depots. However, for the solved instances, the average gap reported by this model is 84.47%. On the other hand, the Wang's model solved a similar number of instances (but it provided feasible solutions for instances up to 100 customers and four depots), reporting an average gap of 79.32.%. Regarding our formulations, the model F2 could solve to optimality all but eight instances, for which the reported gap is on average less than 1% and always below 12.20%. The first model F1 solves the majority of the instances to optimality. For seven cases, the model provides feasible solutions with an optimality gap below 14.02% and for three instances no feasible solution is found.

Table 7 reports the metaheuristic results for the instances with reduced fleet. Clearly, the fleet size decrease makes the problem less computationally tractable as confirmed by the Gurobi average solution time (2881 s) which is almost twice the average solution time reported in Table 5. Of course, this also affects the heuristic solution time which is higher, but always less than 9 min.

Both the splitting and genetic algorithms provide optimal solutions for instances up to 25 customers. Again, the random mechanism provides solutions with slightly lower gaps (6.34% versus 6.48%). Concerning the speed up ratio, the genetic algorithm without the random mechanism is faster.

To sum up, for small instances the proposed metaheuristics provide optimal solutions in neglectable solution time and for larger instances with reasonable fleet size, we obtain quite good feasible solutions with low solution times, but still in some cases the relative gap is quite high.

6 Conclusions

In this study, we introduced the MDk-TRP, generalizing the k-TRP to the multi-depot case, for which we presented two effective mathematical models based on a layered network structure. We investigated the viability of solving the mathematical models by using off-the-shelf software. With respect to other two formulations adapted from the

multi-depot CCVRP, the efficiency of the proposed models, in terms of both solution and time, is very satisfactory. In particular, the formulations solved to optimality instances up to 240 customers within two hours, and found feasible solutions for instances of up to 249 customers. A population-based metaheuristic was designed, including a splitting heuristic for generating good initial solutions and a local search.

The algorithm provided feasible solutions for the instances that could not be solved by any of the models in competitive computational time.

Some interesting extensions of this paper can be investigated. For instance, since we observed that the LP relaxation of the proposed formulations provide good lower bounds, this can be a good starting point to develop an exact or a matheuristic approach. Another research stream that is worth following is the incorporation of stochasticity into model parameters, such as travel time uncertainty or a random presence of customers.

Acknowledgements This research has been partially funded by the Fomento a la Investigación de la Universidad Panamericana [grant number: UP-CI-2021-GDL-04-ING].

References

1. Ángel-Bello, F., Cardona-Valdés, Y., Álvarez, A.: Mixed integer formulations for the multiple minimum latency problem. Oper. Res. **19**(2), 369–398 (2019)
2. Bianco, L., Mingozzi, A., Ricciardelli, S.: The traveling salesman problem with cumulative costs. Networks **23**(2), 81–91 (1993)
3. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The minimum latency problem. In: Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing, pp. 163–171 (1994)
4. Bruni, M., Beraldi, P., Khodaparasti, S.: A fast heuristic for routing in post-disaster humanitarian relief logistics. Transp. Res. Proc. **30**, 304–313 (2018)
5. Bruni, M.E., Beraldi, P., Khodaparasti, S.: A hybrid reactive grasp heuristic for the risk-averse k-traveling repairman problem with profits. Comput. Oper. Res. **115**, 104854 (2020)
6. Chu, F., Labadi, N., Prins, C.: A scatter search for the periodic capacitated arc routing problem. Eur. J. Oper. Res. **169**(2), 586–605 (2006)
7. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Netw. Int. J. **30**(2), 105–119 (1997)
8. Fakcharoenphol, J., Harrelson, C., Rao, S.: The k-traveling repairmen problem. ACM Trans. Algo. (TALG) **3**(4), 40-es (2007)
9. Fischetti, M., Laporte, G., Martello, S.: The delivery man problem and cumulative matroids. Oper. Res. **41**(6), 1055–1064 (1993)
10. Fourer, R., Gay, D.M., Kernighan, B.W.: A modeling language for mathematical programming. Manage. Sci. **36**(5), 519–554 (1990)
11. Gaur, D.R., Mudgal, A., Singh, R.R.: Improved approximation algorithms for cumulative vrp with stochastic demands. Discret. Appl. Math. **280**, 133–143 (2020)

12. Gaur, D.R., Singh, R.R.: Cumulative vehicle routing problem: a column generation approach. In: Conference on Algorithms and Discrete Applied Mathematics, pp. 262–274. Springer (2015)
13. Gaur, D.R., Singh, R.R.: A heuristic for cumulative vehicle routing using column generation. Disc. Appl. Math. **228**, 140–157 (2017)
14. Gurobi Optimization, L.: Gurobi optimizer reference manual. gurobi optimization inc (2020)
15. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
16. Jothi, R., Raghavachari, B.: Approximating the k-traveling repairman problem with repairtimes. J. Disc. Algo. **5**(2), 293–303 (2007)
17. Kara, İ., Kara, B.Y., Yetiş, M.K.: Cumulative vehicle routing problems. In-Teh (2008)
18. Ke, L., Feng, Z.: A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **40**(2), 633–638 (2013)
19. Lalla-Ruiz, E., Voß, S.: A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem. Optim. Lett. **14**(3), 671–691 (2020)
20. Lucena, A.: Time-dependent traveling salesman problem-the deliveryman case. Networks **20**(6), 753–763 (1990)
21. Luo, Z., Qin, H., Lim, A.: Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. Eur. J. Oper. Res. **234**(1), 49–60 (2014)
22. Lysgaard, J., Wøhlk, S.: A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. Eur. J. Oper. Res. **236**(3), 800–810 (2014)
23. Méndez-Díaz, I., Zabala, P., Lucena, A.: A new formulation for the traveling deliveryman problem. Disc. Appl. Math. **156**(17), 3223–3237 (2008)
24. Mladenović, N., Urošević, D., Hanafi, S.: Variable neighborhood search for the travelling deliveryman problem. 4OR **11**(1), 57–73 (2013)
25. Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F., Herazo-Padilla, N.: A literature review on the vehicle routing problem with multiple depots. Comput. Indus. Eng. **79**, 115–129 (2015)
26. Ngueveu, S.U., Prins, C., Calvo, R.W.: An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **37**(11), 1877–1885 (2010)
27. Nucamendi-Guillén, S., Angel-Bello, F., Martínez-Salazar, I., Cordero-Franco, A.E.: The cumulative capacitated vehicle routing problem: new formulations and iterated greedy algorithms. Exp. Syst. Appl. **113**, 315–327 (2018)
28. Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., Moreno-Vega, J.M.: A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem. J. Oper. Res. Soc. **67**(8), 1121–1134 (2016)
29. Ozsoydan, F.B., Sipahioglu, A.: Heuristic solution approaches for the cumulative capacitated vehicle routing problem. Optimization **62**(10), 1321–1340 (2013)
30. Perboli, G., Brotcorne, L., Bruni, M.E., Rosano, M.: A new model for last-mile delivery and satellite depots management: the impact of the on-demand economy. Transp. Res. Part E Logist. Transp. Rev. **145**, 102184 (2021)
31. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**(12), 1985–2002 (2004)
32. Prins, C., Lacomme, P., Prodhon, C.: Order-first split-second methods for vehicle routing problems: a review. Transp. Res. Part C Emerg. Technol. **40**, 179–200 (2014)
33. Ramos, T.R.P., Gomes, M.I., Póvoa, A.P.B.: Multi-depot vehicle routing problem: a comparative study of alternative formulations. Int. J. Log. Res. Appl. **23**(2), 103–120 (2020)
34. Ribeiro, G.M., Laporte, G.: An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **39**(3), 728–735 (2012)
35. Rivera, J.C., Afsar, H.M., Prins, C.: A multistart iterated local search for the multitrip cumulative capacitated vehicle routing problem. Comput. Optim. Appl. **61**(1), 159–187 (2015)
36. Rivera, J.C., Afsar, H.M., Prins, C.: Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. Eur. J. Oper. Res. **249**(1), 93–104 (2016)
37. Salehipour, A., Sörensen, K., Goos, P., Bräysy, O.: Efficient grasp+ vnd and grasp+ vns metaheuristics for the traveling repairman problem. 4OR **9**(2), 189–209 (2011)
38. Sharma, R., Saini, S.: Heuristics and meta-heuristics based multiple depot vehicle routing problem: a review. In: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 683–689. IEEE (2020)

39. Sze, J.F., Salhi, S., Wassan, N.: The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: an effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. Transp. Res. Part B Methodol. **101**, 162–184 (2017)
40. Talarico, L., Meisel, F., Sörensen, K.: Ambulance routing for disaster response with patient groups. Comput. Oper. Res. **56**, 120–133 (2015)
41. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)
42. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. **40**(1), 475–489 (2013)
43. Vieira, Y.E.M., de Mello Bandeira, R.A., da Silva Júnior, O.S.: Multi-depot vehicle routing problem for large scale disaster relief in drought scenarios: the case of the brazilian northeast region. Int. J. Disas. Risk Reduct. **58**, 102193 (2021)
44. Wang, X., Choi, T.M., Li, Z., Shao, S.: An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem. IEEE Trans. Syst. Man Cybern. Syst. **50**(12), 4948–4958 (2019)
45. Wei, X., Qiu, H., Wang, D., Duan, J., Wang, Y., Cheng, T.: An integrated location-routing problem with post-disaster relief distribution. Comput. Indus. Eng. **147**, 106632 (2020)