

Hybrid fleet capacitated vehicle routing problem with flexible Monte–Carlo Tree search

Carmen Barletta, Wolfgang Garn, Christopher Turner & Saber Fallah

To cite this article: Carmen Barletta, Wolfgang Garn, Christopher Turner & Saber Fallah (2022): Hybrid fleet capacitated vehicle routing problem with flexible Monte–Carlo Tree search, International Journal of Systems Science: Operations & Logistics, DOI: [10.1080/23302674.2022.2102265](https://doi.org/10.1080/23302674.2022.2102265)

To link to this article: <https://doi.org/10.1080/23302674.2022.2102265>



© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 03 Aug 2022.



Submit your article to this journal [↗](#)



Article views: 292







View related articles [↗](#)



View Crossmark data [↗](#)

Hybrid fleet capacitated vehicle routing problem with flexible Monte–Carlo Tree search

Carmen Barletta ^{a,b}, Wolfgang Garn ^a, Christopher Turner ^a and Saber Fallah ^b

^aThe Surrey Business School, University of Surrey, Guildford, Surrey, UK; ^bDepartment of Mechanical Engineering Sciences, Connected Autonomous Vehicles Lab, University of Surrey, Guildford, Surrey, UK

ABSTRACT

The rise in EVs popularity, combined with reducing emissions and cutting costs, encouraged delivery companies to integrate them in their fleets. The fleet heterogeneity brings new challenges to the Capacitated Vehicle Routing Problem (CVRP). Driving range and different vehicles' capacity constraints must be considered. A cluster-first, route-second heuristic approach is proposed to maximise the number of parcels delivered. Clustering is achieved with two algorithms: a capacitated k -median algorithm that groups parcel drop-offs based on customer location and parcel weight; and a hierarchical constrained minimum weight matching clustering algorithm which considers EVs' range. This reduces the solution space in a meaningful way for the routing. The routing heuristic introduces a novel Monte–Carlo Tree Search enriched with a custom objective function, rollout policy and a tree pruning technique. Moreover, EVs are preferred over other vehicles when assigning parcels to vehicles. A Tabu Search step further optimises the solution. This two-step procedure allows problems with thousands of customers and hundreds of vehicles to be solved accommodating customised vehicle constraints. The solution quality is similar to other CVRP implementations when applied to classic VRP test-instances; and its quality is superior in real-world scenarios when constraints for EVs must be used.

ARTICLE HISTORY

Received 29 March 2022
Accepted 11 July 2022

KEYWORDS

Routing; MCTS; clustering; optimisation; CVRP

1. Introduction

The transportation sector is one of the largest contributors to greenhouse gas (GHG) emissions in the USA. Medium and heavy-duty trucks are responsible for the 23% of the emissions (United States Environmental Protection Agency, 2020). With growing online retail competition and a shift towards one-day delivery, vehicle pollution is increasing, and roads are becoming even more congested. Given the global concern related to the CO₂ emissions, the need to move away from Internal Combustion Engines (ICEs) or progressively reduce their usage, is becoming a priority. A gradual shift towards the usage of the Electric Vehicles (EVs) powered by renewed energy sources could help reducing GHG emissions by 80–95% by 2050 in Europe (European Environment Agency, 2016).

In this paper, we propose a flexible solution for scheduling parcels' delivery using a hybrid fleet of vehicles. This solution aims to reflect a real case scenario, where a transportation company has different types of vehicles, ICEs and EVs, with variable capacities, available at their depot and would like to effectively maximise the

usage of EVs for delivering its parcels, but also minimise the travelled distance while maximising the number of visited customers. This paper provides the following contributions:

- Formalisation of a two-step clustering technique to group customers based on different multiple constraints;
- Formalisation of the Monte Carlo Tree Search (MCTS) applied to the Hybrid Fleet-Capacitated Vehicle Routing problem (HF-CVRP);
- Application of a flexible MCTS to solve the HF-CVRP combined with a cluster-first route-second approach;
- Validation of the proposed solution through extensive benchmarking.

This paper is organised as follows: Section 2 gives a general problem overview and formulation. Section 3 offers a brief literature review. Section 4 describes the proposed approach in details. Section 5 provides an overview of the experimental results. Section 6 contains the conclusions and proposed future research directions.

2. Problem definition and formulation: HF-CVRP

The problem of minimising transportation costs when visiting a set of customers, by means of multiple routes starting and ending at the depot, is usually represented as a Vehicle Routing Problem (VRP). Many extensions of this problem have been proposed over the past years. One of the most studied problems is the capacitated VRP (CVRP), where vehicles have capacity constraints (Beresneva & Avdoshin, 2018; De Jaegere et al., 2014).

A variant of the CVRP, the HF-CVRP is analysed in this work. The hybrid fleet is composed by both vehicles with ICEs and EVs that can have different capacities. The HF-CVRP accounts for non-homogeneity in the fleet of vehicles. Given a set of vertices representing customer locations, a hybrid fleet, and a depot, the HF-CVRP seeks a set of vehicle routes with minimum distance each of which starts at the depot, visits a set of customers, and returns to the depot without exceeding the EV's driving range and the vehicles' capacities. All the vehicles start their trip with a full tank or charged battery and the EV's are not allowed to recharge their batteries until they complete their deliveries and return to the depot. When assigning vehicles to routes EVs are preferred over those with internal combustion engines (ICEs).

Before introducing the problem in detail an overview of the used symbols is provided in Table 1.

The problem is modelled with a complete directed graph $G = (N, A)$ where N is the set of nodes (customers)

of the graph identifying the depot and delivery locations for the parcels. A is the set of arcs (e.g. road links), where $a_{ij} \in A$ is the arc between node i and node j with distance d_{ij} . $D = (d_{ij}) \in \mathbb{R}_+^{N \times N}$ is the corresponding distance matrix.

Let n be the number of parcels to deliver and m the mixed fleet size, with $m \leq n$. The set of m vehicles that composes the hybrid fleet is made up of a set of EVs $E = \{1, \dots, h\}$ and the internal combustion engine vehicles $F = \{h + 1, \dots, m\}$. Hence, all vehicles are in $K = E \cup F$. The vehicles' load capacities are $C = \langle c_1, \dots, c_m \rangle$. The driving range of each vehicle is defined as $R = \langle r_1, \dots, r_m \rangle$. Each customer parcel $i \in \{1, \dots, n\}$ has a delivery node at location (x_i, y_i) with demand w_i (aka. weight where $W = \langle w_0, w_1, \dots, w_n \rangle$). The depot (i.e. node zero) is located at (x_0, y_0) .

Usually, the objective for the CVRP is to minimise the travelled distance:

$$f(X) = \sum_{i,j \in N, k \in K} d_{ij} x_{ijk}, \quad (1)$$

where x_{ijk} is one if arc a_{ij} is used by vehicle k and zero otherwise. However, in the HF-CVRP it is 'desirable' to maximise the utilise of EVs. The EVs' utilisation can take on various forms such as number of vehicles used, number of parcels delivered, travelled distance. We discuss this in more detail in Section 4.2.2.

The CVRP has the constraints that each customer has to be visited once and only once. For each node (with the exception of the depot) there must be a flow equality. That means, a vehicle k entering node j must leave it again:

$$\sum_{i=0}^n x_{ijk} = \sum_{i=0}^n x_{jik}, \quad j \in \{1, \dots, n\}, k \in K. \quad (2)$$

In the here presented HF-CVRP, we will allow the scenario, that some parcels are not delivered. Equation (2) is still valid in that case. Let y_{jk} represent the decision variable whether a parcel has been delivered. That means, if vehicle k delivered parcel j then $y_{jk} = 1$ otherwise zero. For the previous equation to work, we need to make sure that every node is entered:

$$\sum_{k \in K} \sum_{i \in N} x_{ijk} = 1, \quad j \in N \setminus \{0\}. \quad (3)$$

In the CVRP every vehicle needs to leave the depot:

$$\sum_{j=1}^n x_{0jk} = 1, \quad k \in K. \quad (4)$$

In the introduced HF-CVRP, we will permit that not all vehicles are used. Hence, the above equation will use a subset of K .

Table 1. Overview of symbols used for the HF-CVRP.

Indices & Cardinalities	
i	from location (customer) or general node index
j	to customer number (i.e. parcel identifier) or general index
k	Vehicle identifier
m	Number of vehicles
n	Number of parcels (customers)
Decision variables	
x_{ijk}	Decide whether to visit customer j with vehicle k coming from i
y_{jk}	One, if a customer j has received a parcel by vehicle k ; otherwise zero
u_i	Numbering variable for node i
Variables & Matrices	
a_{ij}	Arc from node i to node j
d_{ij}	Distance of arc a_{ij}
c_k	Capacity of vehicle k
r_k	Driving range of vehicle k
w_i	Parcel i weight (i.e. demand of customer i)
D	Distance matrix
Sets & Sequences	
N	Set of all customers (i.e. parcels and nodes) including depot (node zero)
A	Set of arcs (e.g. road links)
E	Set of electric vehicles $\{1, \dots, h\}$
F	Set of vehicles with internal combustion engines $\{h + 1, \dots, m\}$
C	Sequence of capacities
R	Sequence of driving ranges

Each vehicle must adhere to a capacity constraint:

$$\sum_{j=1}^n y_{jk} w_j \leq c_k, \quad k \in K. \quad (5)$$

In case not all vehicles are needed, adapt Equation (5) by using a subset of K .

For the HF-CVRP the range constraints need to be considered:

$$\sum_{i,j \in N} d_{ij} x_{ijk} \leq r_k, \quad k \in K. \quad (6)$$

Again, it may be required to use a subset of K if not all vehicles are used.

The Miller–Tucker–Zemlin (MTZ) constraints are a convenient way to eliminate subtours (Yuan et al., 2020). Here, u_i is a ‘numbering’-variable for node i , which is greater than zero if the node has been visited. The numbering prevents the return to lower numbers, hence, preventing cycles. u_i also shows the sequence of tours. The following two equations reflect this:

$$u_j - u_i \geq w_j - c_k(1 - x_{ijk}), \quad i, j \in V \setminus \{0\}, \quad i \neq j, \quad k \in K, \quad (7)$$

$$w_i \leq u_i \leq c_k, \quad i \in V \setminus \{0\}, \quad k \in K. \quad (8)$$

3. Related work

The VRP is one of the most studied combinatorial optimisation problems and many variants of the problem have arisen since it was proposed by Dantzig and Ramser (1959). An important variant of the CVRP is the *heterogeneous* CVRP, where the fleet is composed by different types of vehicles. The work by Baldacci et al. (2010) provides a review on exact algorithms for VRP, with a focus on CVRP and on heterogeneous VRPs. Among some of the related works there is the one by Laporte et al. (1985), that first introduced the idea of considering the CVRP with distance constraints. Several authors have provided competitive tabu search heuristics for the heterogeneous fleet VRP (HVRP) such as Brandão (2011), Lai et al. (2016), Meliani et al. (2019) and found new best-known solutions (BKSs). The here proposed algorithm includes a tabu search step as well. Amongst the many important heuristics applied to solve the HVRP, the column generation approach used by Choi and Tcha (2007) is notable, which outperformed other methods for test-instances up to 100 customers. Roughly at the same time (Li et al., 2007) used the record-to-record travel algorithm to solve the HVRP. This algorithm was able to find solutions for test-instances with 360 customers with accuracies close to the BKSs.

Several years later, the evidence hardens that the iterated local search (ILS) heuristic is one of the fastest algorithms delivering high-quality solutions (see Penna et al., 2013; Uchoa et al., 2017) even for large test-instances of the CVRP.

An important addition to the existing heterogeneous VRPs was given in Erdoğan and Miller-Hooks (2012) formulation, where a green VRP included alternative-fuel vehicles that could be refuelled at specific stations. The concept of a fleet composed by EVs associated with time windows deliveries and recharging stations was proposed by Schneider et al. (2014) known as Electric-VRP with Time Windows (E-VRPTW).

With regards to mixed fleet, Goeke and Schneider (2015) addressed and proposed a load-dependant energy consumption model to solve the E-VRPTW with a mixed fleet composed by a single type of EV and ICE.

Hiermann et al. (2019) presented an E-VRP that combines conventional, plug-in hybrid, and EVs and solved it using a hybrid genetic algorithm based on layered route evaluation procedures.

The HF composed by different types of EVs and different types of ICEs, as it is considered in this work, has been tackled also by Lebeau et al. (2015). In their work, they also introduced an energy consumption model, however, their experimental results considered only instances with up to 25 parcels.

It should be noted that variations of the Iterated Local Search (ILS) have been found to outperform most other VRP heuristics (see Uchoa et al., 2017). For instance, Penna et al. (2013) and Subramanian et al. (2012) introduced one of the first ILS metaheuristics with a random neighbourhood ordering (RVND) in the local search phase for the Heterogeneous Fleet Vehicle Routing Problem.

For deep reviews, consider the following references Gharaei, Karimi, et al. (2021), Sbair and Krichen (2022), Gharaei, Hoseini Shekarabi, et al. (2021), Zhuang et al. (2022), Askari et al. (2021), Taleizadeh et al. (2022), Vidal (2022), Gharaei, Amjadian, et al. (2021), Amjadian and Gharaei (2021) and Gharaei et al. (2021).

4. Solution to the HF-CVRP

The proposed algorithm follows the cluster-first, route-second approach. In this approach, the HF-CVRP is decomposed into smaller subproblems by first clustering customers into groups and then assigning clusters to vehicles. Note however that our implementation differs from the classical cluster-first, route-second heuristics. In the classical formulation, each vehicle is associated with a single cluster (e.g. see Comert et al., 2018). Motivated, by typical geographical circumstance (e.g. hamlets, villages,

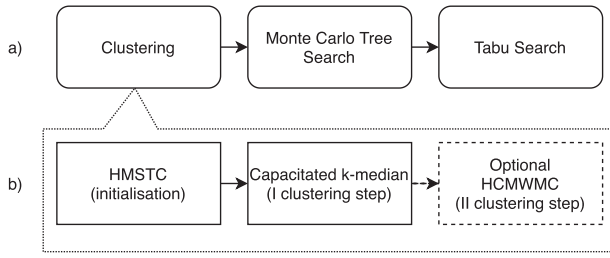


Figure 1. General algorithm structure. (a) Macro steps of the cluster-first, route-second approach with a final Tabu Search optimisation step. (b) The internal clustering steps.

towns) we propose to cluster customers first irrespective of the number of vehicles. That means, we propose the concept of an *aggregated customer*. The number of clusters built is expected to be greater than the number of available vehicles and therefore each vehicle can be assigned to more than one cluster. However, it could happen that a cluster exceeds the capacity of a single vehicle, and therefore each vehicle can be assigned to more than one cluster. Indeed, by clustering customers, the routing step can be used to solve the HF-CVRP on a smaller number of aggregated customers. A final optimisation step is then added to compute the specific route for a particular vehicle by minimising the route linking all the customers belonging to all the clusters assigned to that vehicle.

Figure 1 highlights the three main steps of the proposed algorithm. Figure 1(a) shows the macro steps of the cluster-first, route-second approach with a final Tabu Search optimisation step. Figure 1(b) shows the internal clustering steps composed by a Hierarchical Minimum Spanning Tree Clustering (HMSTC) used for the centroids' initialisation, followed by a Capacitated k -median clustering and an optional Hierarchical Constrained Minimum Weight Matching Clustering (HCMWMC) that is executed only on clusters violating the driving range constraints.

The procedure is described more in details in the subsequent sections, which will make use of the nomenclatures shown in Table 2.

4.1. Clustering

The clustering step allows for the aggregation of parcels in such a way that they can be represented as a single entity while computing the routes during the MCTS. The clustering phase plays an important role in the solution finding process, as it helps to speed up the computation by reducing the dimensionality of the problem at hand. This is similar to focusing on relevant decision variables only. Appropriate constraints related to the vehicles' characteristics are required in order to encourage solutions

that successfully deliver all parcels and exploit the EVs usage. More specifically, it must be ensured that for each cluster the following constraints are fulfilled:

- (1) *Capacity constraints*, the total cluster's capacity demand does not exceed a predefined maximum capacity threshold \hat{C} ;
- (2) *Driving range constraints*, the estimated total route length required to deliver all the parcels' locations in a cluster, does not exceed a predefined threshold \hat{R} .

The first constraint is required given that all the vehicles have a maximum load capacity. The second constraint is needed by the fact that each EV should be able to complete all the parcels' deliveries inside a cluster without requiring to recharge its battery. For ICE vehicles it is assumed that their range is either sufficient for the entire tour; or that they can refill on the way without affecting the journey time significantly. To address the fact that the fleet can be heterogeneous the two thresholds are set in a conservative way, that is, the \hat{C} is equal to the minimum load capacity and the \hat{R} is equal to the minimum driving range over all the vehicles. The proposed clustering methodology follows a two-step approach as shown in Figure 1(b) where each step focuses on one of the two constraints. By solving the clustering problem in two separate steps, we avoid using a single integer linear programming formulation that would require an $\mathcal{O}(n^3)$ number of constraints, where n is the number of parcels. Indeed, several preliminary experiments involving linear programming formulation and balanced constraints (Malinen & Fränti, 2014) have shown that exact approaches are infeasible for bigger instances. This finding has led to the creation of the heuristic explained below.

While performing the clustering step the depot is removed from the starting graph and the directed graph is transformed into a fully connected undirected graph where $d'_{ij} = \max\{d_{ij}, d_{ji}\}$ for each $i, j \in V$.

4.1.1. The first clustering step

The first clustering step (Figure 1(b)) is a modified version of the k -means capacitated clustering algorithm described in Geetha et al. (2009). At this stage, parcels are grouped into k clusters with a restricted \hat{C} with the objective of minimising the total assignment cost distance. The proposed solution implements a heuristic approach where parcels are assigned to their closest centroids given that they have the highest priority over the other candidates' members, but some changes have been introduced with respect to the original version.

k -median algorithm is preferred over k -means, as it is less susceptible to outliers, since the median is a robust

Table 2. Nomenclatures of the MCTS for the HF-CVRP.

Var.	Description	Var.	Description
α	Loose upper bound	p	Penalty for using ICE vehicles (dependent on state)
γ	Reward adjustment (dependent on state and delivery status)	\hat{R}	Minimum driving range
C	Degree of exploration	R_i^t	Route of vehicle i at turn t
\hat{C}	Minimum load capacity	s^t	State of search (i.e. all routes) at turn t
δ	Depot	S	All possible states
d'_{ij}	Distance between parcel position i and centroid-parcel j	t	Intermediate turn
D'	Matrix of distances to centroids	T	Current turn
$D_E(s^t)$	Route travelled by EVs in state s^t	U	Number of selected edges
\bar{e}_{ij}	Edge between cluster-node i and j	v	Area in Euclidian plane
\bar{E}	Edges (cluster)	\bar{V}	Vertices (subset of \hat{N})
θ	Penalisation weight	v_i	Frequency of visits to node i
k	Number of clusters	\bar{w}	Weights (cluster)
k_{\min}	Minimum number of clusters	\bar{w}_i	Demand at node i
μ	Negative reward parameter (dependent on state)	X	Decision matrix of nodes assigned to clusters
\hat{N}	Variable set of nodes (nodes in cluster)	\bar{X}_j	Average reward of the subtree rooted in node j
\hat{n}	Number of parcels in cluster		

statistic of central tendency, while the mean is not (Venables & Ripley, 2013) (real representative centroids need to be considered). Indeed, since the clusters centroids are real parcel node representatives, the real distances (instead of the Euclidean one) can be considered to assign nodes to clusters. Moreover, during the initialisation step, centroids are chosen with a HMSTC algorithm (Figure 1(b)) as described in Zhou et al. (2011) instead of a random selection. This allows a more uniform centroids distribution. Additionally, when performing the capacitated clustering algorithm, the priority of a parcel i being added to a cluster j is inversely proportional to d'_{ij} where d'_{ij} is the distance between the parcel position i and the position of the centroid-parcel j . The last difference is related to the computation of the number of needed clusters k . Instead of computing $k = \lceil \frac{\sum_{i=0}^n w_i}{\hat{C}} \rceil$, which does not guarantee that all the parcels can be assigned to one cluster, a custom procedure has been devised. It takes inspiration from the greedy approximation algorithm, which solves the unbounded knapsack problem (Dantzig, 1957). It estimates a worst case scenario where all the parcels with highest demand belong to the same cluster and computes the maximum number of parcels that would fit in without violating the capacity constraints. The algorithm also guarantees the detection of singular nodes (i.e. nodes that cannot be matched with any other set of nodes) and adjusts the value of k accordingly. Algorithm 1 captures these ideas.

The input for the algorithm is a set of nodes N , the demand vector w , a capacity limit for the cluster \hat{C} , and a minimum number of clusters k_{\min} . First, parcels are sorted by decreasing order of demand ($w_0 \geq w_1 \geq \dots \geq w_n$). Then they are added to the same *cluster* set as long as the capacity limit is not exceeded. However, if the first two parcels exceed the capacity constraint, then the item in *cluster* is added to the set of *singularNodes* and the *cluster*

Algorithm 1: ComputeK(N, w, k_{\min}, \hat{C})

Input: set of nodes N , demand w , capacity \hat{C} , minimum number of clusters k_{\min}
Result: The number of clusters k
 $cluster \leftarrow \emptyset$; $singularNodes \leftarrow \emptyset$;
 // initialisations
 N.SortDecreasingDemand(w)
for $e \in N$ **do**
 if $cluster.Fits(e, \hat{C})$ // i.e. $w_1 + \dots + w_e \leq \hat{C}$
 then
 $cluster.Add(e)$
 else if $|cluster| == 1$ // i.e. $w_1 + w_2 > \hat{C}$
 then
 $singularNodes.Add(cluster.First())$
 $cluster \leftarrow \emptyset$
 $cluster.Add(e)$
 else
 $k \leftarrow \lceil \frac{|N| - |singularNodes|}{|cluster|} \rceil$
 $k \leftarrow k + |singularNodes|$
return $\max\{k_{\min}, k\}$

is emptied. Finally, the number of clusters k is equal to the number of parcels excluding the singular ones divided by the number of parcels in the ‘worst case’ *cluster* and incremented by the number of parcels considered singular. To avoid k is set to zero the maximum between a given minimum number of clusters k_{\min} and the computed value k is returned.

4.1.2. The second clustering step

After performing the first clustering algorithm, each cluster built is ensured to fulfil the capacity constraints, but no guarantee is given for the *driving range constraints*.

To overcome this problem, the route needed to deliver all the parcels in a cluster is estimated by using a continuous approximation model; and the clusters where the estimate exceeds the predefined \hat{R} are fed to the second clustering step (Figure 1(b)). By considering the continuous approximation formula for the Travelling Salesman Problem (TSP) proposed by Applegate et al. (2011), the estimate can be computed as \sqrt{nv} where n is the number of parcels in a cluster and v is its bounded plane region area (Garn, 2020, 2021).

This second step considers only the driving range constraints and is implemented as a Hierarchical Constrained Minimum Weight Matching clustering algorithm (HCMWMC) (Figure 1(b)). The goal is to consider all the \hat{N} cluster's nodes as separate entities and progressively group them in sub-clusters. The pseudocode in Algorithm 2 implements the HCMWMC routine.

Algorithm 2: HCMWMC(\hat{N} , D' , \hat{R})

Result: The \hat{N} nodes are grouped in clusters without violating the \hat{R} constraint

- 1: **do**
 - 2: $X = \text{SolveAssignment}(\hat{N}, D', \hat{R})$
 - 3: $\hat{N} = \text{UnifyNodes}(X)$
 - 4: $D' = \text{UpdateDistances}(\hat{N})$
 - 5: $\hat{N} = \text{RemoveOutliers}(D', \hat{R})$
 - 6: **while** $X \neq \text{null}$ and $|\hat{N}| > 1$
 - 7: **return** \hat{N}
-

The \hat{N} nodes are taken as input together with the symmetric distance matrix D' and the \hat{R} parameter. First, all the nodes that exceed the \hat{R} with respect to all the other nodes are removed from the set \hat{N} . These nodes are called outliers and their removal procedure is represented by the function *RemoveOutliers*. The function *SolveAssignment* solves the optimisation assignment problem described in linear problem (LP) formulation (Equations (9)–(14)) and returns an assignment matrix $X = x_{ij} \in \mathbb{B}^{\hat{N} \times \hat{N}}$, where $x_{ij} = 1$ if node i and node j can be grouped together without violating the driving range constraints. The function *UnifyNodes* groups nodes that have been matched together and returns the new set of grouped nodes \hat{N} .

In accordance to the grouping, the distance matrix D' is updated by the *UpdateDistances* function and the outliers are again found and removed from the set \hat{N} . The procedure is repeated until all the nodes have been grouped together in the best case or there is no further feasible matching.

The optimisation problem described in Equations (9)–(14) is a modified version of the minimum weight matching (Kolmogorov, 2009) and its objective is to select a subset of edges minimising the total distance. Constraint (10) ensures that for each node $i \in \hat{N}$ at most one edge is selected, while constraint (11) enforces that if edge (i, j) is selected then also the symmetric edge (j, i) is selected as well. Constraint (12) ensures that for each node i the weight of the selected edge does not exceed the \hat{R} parameter and finally constraint (13) ensures that the number of selected edges is equal to U , where $U = |\hat{N}|$ if \hat{N} is even otherwise $U = |\hat{N}| - 1$, so this allows to match nodes in pairs.

$$\min_x z = \sum_{i \in \hat{N}} \sum_{j \in \hat{N}} d_{ij} x_{ij} \quad (9)$$

$$\sum_{j \in \hat{N}, j \neq i} x_{ij} \leq 1 \quad \forall i \in \hat{N} \quad (10)$$

$$x_{ij} = x_{ji} \quad \forall (i, j) \in \hat{N} \times \hat{N} \quad (11)$$

$$\sum_{j \in \hat{N}} x_{ij} d_{ij} \leq \hat{R} \quad \forall i \in \hat{N} \quad (12)$$

$$\sum_{i \in \hat{N}} \sum_{j \in \hat{N}, j \neq i} x_{ij} = U \quad (13)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in \hat{N} \times \hat{N} \quad (14)$$

After the clustering steps, the final number of clusters is smaller than the initial number of parcels N . Using these clusters, a new graph is built. The graph $\bar{G} = (\bar{V}, \bar{E})$ is a directed graph where each node \bar{V} represents a cluster of nodes with weights $\bar{W} = \langle \bar{w}_0, \dots, \bar{w}_{\hat{n}} \rangle$ equal to the sum of the nodes' weights within the cluster. Each edge $\bar{e}_{ij} \in \bar{E}$ between node (cluster) i and j is equal to the edge of minimum distance among the subset of edges connecting a node in i with a node in j . Moreover, the depot is added back to the graph \bar{G} .

4.2. Monte Carlo Tree search

The MCTS (Browne et al., 2012; Hashimoto et al., 2012) is a heuristic-driven search algorithm. That is a combination of classic tree search implementations alongside reinforcement learning principles (Sutton & Barto, 2018). MCTS is often used in combinatorial games to efficiently explore the solution space and find the most promising moves for a player that lead to win the match. The main advantage of the MCTS is that it has a knowledge-free nature (Mańdziuk, 2010, 2011), i.e. the only domain knowledge required is the ability to distinguish a good solution from a bad one this is why this algorithm is flexible. It became popular when AlphaGo (Silver et al., 2016)

(computer program utilising MCTS) beat a human professional Go player. The MCTS works on the search tree representation of the solution space and efficiently performs searches on it without extensively exploring the tree itself (Coulom, 2009). This is incredibly powerful if applied to problems with a large branching factor. For instance, the number of children at each node of the tree has an average of 250 at Go (Silver et al., 2016) and this makes an exhaustive search infeasible. The MCTS technique has been successfully applied to implement AI systems for different types of games, from connection games (such as *Hex* Arneson et al., 2010) to single player (e.g. *Morpion Solitaire* Cazenave, 2009) and real-time games (e.g. *Ms. Pac-Man* Tong et al., 2011) as pointed out in the survey by Browne et al. (2012). Other applications outside of the field of games, include evaluation of security authentication systems (Tanabe et al., 2009) and production management problems (Lubosch et al., 2018). However, there are few attempts in the literature that aim to apply the MCTS technique to solve the CVRP and to the best of the authors' knowledge no one has attempted to apply MCTS technique to solve the HF-CVRP. The work by Mańdziuk and Nejman (2015) represents one of the first attempts in this field. Mańdziuk and Nejman (2015) assumes that, for a given instance of the problem, a good estimate of the optimal solution is known in advance and this is leveraged in order to compute the objective function. Unfortunately, this is an unrealistic assumption in real case scenarios. According to Edelkamp et al. (2016) MCTS is predicted to show many further advances in a widespread set of industrial relevant applications and with their research they show that this randomised optimisation method is indeed promising and, in some cases, can produce state-of-the-art solutions to challenging problems. In Abdo et al. (2016) the MCTS is used to solve the VRP with pickup and delivery, but the experimental results have been executed only on instances up to 305 orders. The main contribution of this paper is to: formulate the MCTS in terms of CVRP and apply this technique to find good solutions to the HF-CVRP. The use of the MCTS to solve this problem provides a chance to deal with multiple additional constraints and to build a flexible heuristic that can be easily adapted to find solutions to different variations of the same problem.

4.2.1. MCTS algorithm

The MCTS relies on representing the search space of all the possible solutions as a search tree. The root node represents the initial state of the search space. Each node in the search tree represents a state, and directed links to child nodes represent actions leading to following states.

Since building the complete search tree would be a computationally and memory intensive process, the search tree is built incrementally starting from a root without any children. The execution of the MCTS is an iterative process where many trajectories (state-action sequences) are simulated from the current state until a terminal state is reached. The actions in the simulated trajectories are generated using a policy, usually called *rollout policy*. Only initial portions of the trajectories with high evaluations from the performed simulations, are actually added to the current state of the tree. The value of a state-action pair is estimated as the average of the simulated returns from the pair. While the rollout policy is used to select actions starting from a leaf node and therefore outside from the actual tree, the *tree policy* is used to select the actions inside the tree according to the action's values that have been estimated by balancing between *exploration* and *exploitation*.

More precisely, at each iteration of the MCTS, the following steps are executed:

- (1) *Selection*: the tree is explored, using the *tree policy* based on the action values attached to the edges of the tree, starting from the root node and a leaf node of the tree is selected.
- (2) *Expansion*: the children of the selected leaf node are generated, by using unexplored actions that lead from the selected node to its children.
- (3) *Simulation*: starting from the selected node or from one of its newly added children, a simulation is performed until a terminal state is reached. During the simulation, the actions are chosen according to the *rollout policy*.
- (4) *Backpropagation*: the return generated by the simulation is backpropagated up in the tree, starting from the leaf node where the simulation originated. All the action values attached to the edges of the tree traversed by the tree policy are updated.

The *tree policy* selects the actions using the *Upper Confidence Bound for Trees (UCT)*, that is an adaptation of the *Upper Confidence Bound (UCB)* (Justesen et al., 2017).

The UCT gives an estimate on the most promising moves leading to nodes in the tree and moreover it addresses the *exploration vs. exploitation dilemma*: one needs to balance the *exploitation* of the action currently believed to be optimal with the *exploration* of other actions that currently appear suboptimal but may turn out to be superior in the long run (Browne et al., 2012).

Given a parent node i , the UCT value of a child node j , is computed with the following formula:

$$UCT = \bar{X}_j + C \sqrt{\frac{\log v_i}{v_j}}$$

where:

- $C > 0$ is a constant term that controls the degree of exploration;
- \bar{X}_j is the average reward of the subtree rooted in node j
- v_i and v_j are the number of times that node i and j are visited, respectively;

The average reward term \bar{X}_j encourages *exploitation* of subtrees with higher cumulative rewards, while $\sqrt{\frac{\log v_i}{v_j}}$ encourages *exploration* of less visited subtrees. The UCT yields infinite when $v_j = 0$ so that previously unvisited children nodes are assigned the largest possible value and ensure that all children of a node (and therefore the actions leading to those children) are considered at least once before any child is expanded further. However, actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time. This results in a powerful form of *iterated local search* (Browne et al., 2012).

The pseudocode in Algorithm 3 shows the MCTS works by performing an exploration of the search tree starting from the *rootNode* until a computational budget is reached (i.e. number of iterations). At each iteration, the exploration returns a new feasible solution that is checked against the best solution found. The best solution is the one maximising a given reward function.

Algorithm 3: MCTSDriver

Result: Best solution found
 Initialise root node r_0 and *bestSolution*
for $i \leftarrow 0$ **to** N_ITERS **do**
 $solution \leftarrow MCTS(rootNode)$
 if $Reward(solution) > Reward(bestSolution)$
 then
 $bestSolution \leftarrow solution$

The four main steps that compose the MCTS are represented as a pseudocode in Algorithm 4.

MCTS can be implemented with a recursive function that returns when a leaf of the tree has been reached, that is, the node has been fully expanded, or after an *expansion* and *rollout* operation. An input node is *expanded* if it is not a terminal node (*leaf node*). The child node with the highest UCT is selected (*bestChildNode* in Algorithm 4). If the subtree of the best child has not been explored

Algorithm 4: MCTS(node)

Result: A solution found while exploring the search tree
if *IsLeaf(node)* **then**
 $\text{return } node.solution$
 $node.Expand()$
 $bestChildNode \leftarrow node.SelectBestUCTChild()$
if $UCT(bestChildNode) == \infty$ **then**
 $solution \leftarrow Rollout(bestChildNode)$
else
 $solution \leftarrow MCTS(bestChildNode)$
 $UpdateNode(node, solution)$
 // back-propagation
return $solution$

yet, that is, its UCT yields infinite, then a *rollout* is performed starting from that node, otherwise the exploration of the tree continues recursively. When a leaf is reached or a rollout is performed, the final outcome is *back-propagated* to the root of the tree and all the v and \bar{X} values of the nodes that have been visited during the search are updated accordingly.

4.2.2. Formalisation of the MCTS applied to the HF-CVRP

This subsection describes how the HF-CVRP has been modelled in function of the key features required by the MCTS. It is crucial, indeed, to describe which kind of actions MCTS can perform during the search, how a state of the search is defined and what is the objective of the search itself.

The objective of the search is to find a solution to the problem that:

- minimises the total route length;
- maximises the number of parcels delivered;
- maximises the usage of EVs, therefore minimising the CO₂ emissions.

At every MCTS iteration, actions are performed to explore and expand the search tree. A turn represents the transition from one state to another by performing an action. An action involves the assigning to a vehicle the next parcel to deliver. The vehicle that is used depends on the current state of the search. There is always a *dummy action* that can be performed, that is, send the vehicle back to the depot. The root level of the game tree represents the starting state, where all the vehicles are ready to start their journey at the depot and the EVs are fully charged. A leaf of the tree represents the state of the search where all the vehicles have returned to the depot. More formally:

- The route of vehicle i at turn T is $R_i^T = \{\delta^0, n_i^1, n_i^2, \dots, n_i^T\}$ where δ identifies the depot and n_i^t identifies the parcel delivered by vehicle i at turn $t \leq T$. At every turn, only the route of a single vehicle is altered. R defines the set of all the possible routes.
- The state of the search $s^t = \{R_0^t, \dots, R_m^t\}$ identifies the set of routes of each vehicle at turn t . All the possible states of the search tree define the set S .
- The function $nextVehicles : S \rightarrow VE'$ where $VE \subseteq VE'$ identifies the set of vehicles to which a parcel could be allocated to in the next turn.
- $s^0 = \{R_0^0, \dots, R_m^0\}$ with $R_i^0 = \{\delta^0\} \forall i \in m$ is the initial root state of the search.
- If s^T is a terminal state, then $last(R_i^T) = \delta \forall R_i^T \in s^T$, given that $last : R \rightarrow N$ is a function returning the last visited node of a route. This means that all the vehicles must have returned to the depot.
- The set of possible actions is computed by the function $nextActions : (S, V) \rightarrow N'$ where $N' \subseteq N$ that given a vehicle and the current state, it returns a subset of parcels the vehicle V can deliver. The subset N' always includes the *dummy action*.

Each node of the tree is associated to a unique state of the search. The root is associated with state s^0 . The function $Reward : S \rightarrow \mathbb{R}$ in Algorithm 3 computes the utility function of a state s^t . It is defined by the following formula:

$$Reward(s^t) = \frac{\alpha - (D(s^t) + \mu(s^t))}{\alpha} \times \gamma(s^t) \times p(s^t)$$

where, α represents a loose upper bound on the total travelled distance; $D(s^t)$ is the total travelled distance in the state s^t ; $\mu(s^t)$ is a negative reward parameter that estimates the route needed to deliver all the remaining undelivered parcels, if any and $\gamma(s^t) = 1$ if all the parcels are delivered with the routes in s^t , otherwise it is equal to 0.5. These first two factors of the reward function are adapted from the work of Kartal et al. (2016). The term $p(s^t)$ is another penalisation factor, introduced here to discourage solutions that prefer ICEs vehicles over the EVs. It is computed by the following formula:

$$p(s^t) = \theta + (1 - \theta) \frac{D_E(s^t)}{D(s^t)}$$

where $D_E(s^t)$ is the route travelled by EVs in state s^t and $0 \leq \theta \leq 1$ is a constant factor deciding the penalisation weight, the lower is θ the bigger is the penalisation. $\theta = 1$ if there are no EVs in the set of vehicles VE .

Given a node and its associated state s^t , the function $GenerateChildren : S \rightarrow \{S\}$ generates the set of all the possible following states. First of all, the function

$nextVehicles$ is used in order to identify all the available vehicles that can deliver a parcel. These are all the vehicles i that have not completed their trip yet, that is, such that $last(R_i^t) \neq \delta$. One vehicle among these, different from the one that has delivered a parcel in the previous state, is chosen at random. Second, the function $nextActions$ is used to identify all the available parcels that the chosen vehicle can deliver. The function considers only parcels that have not been delivered at the previous state and that satisfy capacity constraints for the chosen vehicle. Indeed, if the chosen vehicle is electric, then the function also checks driving range constraints. By identifying a vehicle and the possible parcels to deliver, the next states can be generated.

In the classical MCTS version, the *rollout policy* is used to generate simulated trajectories and consists in choosing uniformly at random moves until a terminal state is reached. In the proposed version, the rollout is performed by doing a weighted sampling over all the actions returned by the function $nextActions$. Each parcel has a probability of being selected inversely proportional to the distance from the current position of the vehicle. This sampling directs the search towards better solutions as described in Cazenave et al. (2020).

The function $IsLeaf : S \rightarrow B$ where $B = \{0, 1\}$ returns 1 if the state in input is terminal, that is, if all the vehicles have returned to the depot or if none of the vehicles can deliver additional parcels because the driving range constraints or the capacity constraints are violated.

4.2.3. Tree pruning and parallelisation

To speed up the search and find a good solution more efficiently two techniques are employed: a pruning criteria that allows to cut suboptimal branches of the tree and two types of parallelism both at *rollout* and *root* level (Chaslot et al., 2008). The pruning is inspired to the branch and bound technique described in Kartal et al. (2016). The idea is to branch new nodes from a parent node with state s^t , only if the route length of the global best solution found so far, which allocates all the parcels, is bigger than the total route length represented by state s^t plus a lower bound estimate on the route length needed to complete the allocation of s^t . This estimate is the route needed to deliver all the parcels and it is arrived at by continuing to explore the tree from s^t .

The performances of the root level parallelisation have shown promising results (Fern & Lewis, 2011; Soejima et al., 2011). Its principle is to build simultaneously different search trees and synchronise their results after a certain computational budget is reached. In the proposed solution, different search trees are initialised and each of them considers a different vehicle as the first one choosing the parcels to deliver at root state of the

tree. Indeed, since the considered fleet is mixed, this parallelisation allows for the exploration of a more heterogeneous set of solutions. The other level of parallelisation employed is inspired to the *leaf parallelisation* (Chaslot et al., 2008), but instead of performing different rollouts from the same leaf node, in the proposed *rollout parallelisation* multiple playouts are carried out from different leaf nodes simultaneously, provided that they have the same UCT, so that the tree is explored faster.

4.3. Tabu search

The last optimisation step of the algorithm consists in executing a Tabu Search over the best solution found by the MCTS, as shown by Figure 1(a). Tabu Search was initially proposed by Glover (1986) and it is a local search method for combinatorial optimisation. The method performs an exploration of the solution space by moving from a solution sol_t identified at iteration t to the best solution sol_{t+1} found in a subset of the neighbourhood of the solution sol_t . To encourage the exploration of the solution space and avoid becoming stuck in local optima, a memory structure called the tabu list (which saves recently visited best solutions) is used.

The main objective for applying the Tabu Search in this algorithm is to reduce the total route length within each route R_i^T . At this stage, the clusters are removed and each parcel n_i is considered as an independent entity in the route. To avoid breaking capacity and driving range constraints, one Tabu Search for each route is performed and only intra-route moves are allowed, when performing a neighbourhood search. For the first solution strategy a nearest neighbour heuristic (Kizilates & Nuriyeva, 2013) is used. The 2-opt swap moves (Croes, 1958) are the chosen strategy applied to generate the neighbourhood of a solution. A short-memory is used as a tabu list. The output of this step is the final optimised set of routes.

5. Experimental results

To assess the performances and correctness of the proposed algorithm, several experiments have been executed on the CVRP benchmark instances of the Uchoa et al. (2017) dataset. This dataset is composed of 100 instances ranging from 100 to 1000 customers and has been designed in order to provide a more comprehensive and balanced experimental setting which can cover the wide range of characteristics found in real applications. The tests have been run on an Intel Xeon Gold 5120 with 2.2 gigahertz and 192 gigabytes of RAM, running under CentOS 7. For each instance, the proposed algorithm has been run using 28 threads, where four threads have

been used for the *root* level parallelism each using seven threads for the *rollout* level parallelism. The UCT constant C has been set to $\sqrt{2}/1000$. To allow smaller clusters to be built, $\hat{C} = Q/2$, where Q is the vehicles capacity reported by the test instances. The results given by the proposed algorithms are compared not only with the optimal known solution in the literature so far, but also with the results given by Google OR Tools (GORT) Routing engine (Perron & Furnon, 2019), which at the best of the authors' knowledge is the only solver capable of similar problem flexibility and adaptability readily available. The GORT approach to solve the CVRP is mainly done using approximate methods (namely local search), potentially combined with exact techniques based on dynamic programming and exhaustive tree search. The proposed solution and the GORT (version 7.8.7959) testing routine have been implemented in C#. The time execution limit has been set to 30 minutes for all the instances.

The best results found are reported in Appendix 1. Moreover, for the convenience of the reader we provided the BKS and results found in Uchoa et al. (2017) for comparison purposes. The name of the instances have the format X-nA-kB, where A represents the number of points in the instance including the depot, and B is the minimum possible number of routes, calculated by solving a bin-packing problem. The *RoutedRatio* represents the fraction of routed parcels and it is the ratio between the number of routed parcels and the total number of parcels, while the *DistanceRatio* is the ratio between the total route length found and the total route length of the optimal solution reported in Uchoa et al. (2017), and it describes how good the found solution is with the respect to the optimal one. The instances where the *RoutedRatio* < 1 are marked with the symbol (*). For the MCTS algorithm the average *RoutedRatio* is 0.98 with a standard deviation of 0.03, which means that most of the parcels are routed: in 98 out of 100 instances a *RouteRatio* of at least 0.90 has been achieved; out of them 37 achieved a *RoutedRatio* equal to 1. The GORT experiments instead show a *RoutedRatio* = 1 for all the test instances solved in the given time frame, but for 39 out of 100 instances no solution could be found, which are marked with symbol '–' in Appendix 1.

In our solution, the average *DistanceRatio* for the instances where all the parcels have been allocated (cases where *RoutedRatio* = 1) is equal to 1.37 with a standard deviation of 0.26, while GORT achieved an average value of 1.09 with a standard deviation of 0.05. Indeed, our usage of a cluster-first, route-second approach simplifies the exploration of all the possible solutions, at the expenses of a less accurate consideration of the distances between the customers.

At first sight, GORT option may look better, but in reality it shows an "all-in" property: either it finds a nearly optimal solution, or it does not even find a valid allocation. Most of the instances where GORT fails to find a solution are composed by parcels whose weight is close to the maximum vehicles' capacity. Our solution mitigates this issue and it successfully allocates most of the parcels even in these critical scenarios as previously mentioned.

To test the algorithm on a mixed fleet, some of the Uchoa et al. (2017) instances have been modified and the results have been reported in Appendix 2. For these tests, GORT engine has been used also as a solver for the LP (Equations (9)–(14)) of the second clustering step described in Algorithm 2.

For each instance of the form X-nA-kB, the columns in Appendix 2 are the following: R is the driving range assigned to each EV and has been computed by dividing the optimal route length of a specific instance by the minimum number of required vehicles. Column EV_s represents the number of EVs in the fleet and it is equal to the minimum number of required vehicles divided by two ($\lceil B/2 \rceil$). Column $ICEs$ reports the number of ICEs for each instance and it is equal to the number of EVs plus an additional 10% ($\lceil B/2 \rceil + \lceil B/2 \rceil 0.1$). The *NotUsedCars* represent the number of ICEs vehicles that have not been used in the final solution. The *EV Usage* column is the ratio between the average of the total route travelled by the EVs and their driving range R and indicates how much EVs have been exploited compared to their possibilities. All the vehicles have been considered to have the same capacity reported in the original instances. The average *EV Usage* is equal to 0.91 with a standard deviation of 0.04 and shows that EVs are used efficiently in the solution. One of the instances reports a *NotUsedCars* = 1 which gives evidence that in some cases the algorithm is able to find the minimum number of required vehicles to deliver all the parcels itself.

The approach proposed in this work to solve the HF-CVRP shows how EVs can effectively be integrated in a fleet of ICEs vehicles and efficiently be used to deliver parcels. This can help fleet managers to progressively introduce EVs in their fleet and therefore measure their cost saving and emission reduction before making a full transition from an ICE fleet to an EV fleet. Furthermore, it allows managers to take the driving range of the EVs into account.

Another advantage of the approach is the consideration of low emission zones (or any special areas). The proposed cluster-first route-second approach offers flexibility of accommodating constraints. For instance, London's Ultra Low Emission Zone (ULEZ) is an area, where most vehicles that do not meet the ULEZ emission standards

must pay a daily charge to drive inside the zone. Therefore, delivering parcels that are located in the ULEZ by using EVs can help reduce delivery costs, since EVs are exempt from paying the daily fee. This additional constraint can be easily added to the proposed algorithm to obtain solutions that reduce the delivery costs. Indeed, parcels inside the boundaries of a ULEZ can be clustered together in a preliminary step, using the proposed clustering algorithm. Then the MCTS reward function can be adapted with an additional term so that solutions where clusters of parcels located in the ULEZ are assigned to EVs, are preferred over the others.

6. Conclusion and future work

In this work we have proposed a flexible approach to solve the HF-CVPR, to optimise the routing of a mixed fleet of vehicles composed by ICE vehicles and EVs. Contrary to other approaches we have not introduced the recharging stations given that the time to recharge the batteries for an EV could significantly delay the parcels' delivery. We implemented a new type of cluster-first, route-second approach. For the clustering part we have introduced an incremental approach to deal with all the constraints related to the driving range and the vehicles' capacities. Two main clustering techniques have been implemented: a modified version of the k -median and a Hierarchical Constrained Minimum Weight Matching. These two techniques allowed to split the customers by reducing the dimensionality of the problem taking into account not only the number of available vehicles but also the vehicles' constraints and customers distribution.

For the routing, MCTS was formalised and applied to the HF-CVRP. Customisation in the form of the reward function involving a modified rollout policy along with pruning techniques, has been introduced. The reward function allows the algorithm to prefer solutions where EVs are used more than the ICEs and at the most of their possibilities.

The algorithm performs well with instances where the customers' distribution follows a clustered pattern and proves to be effective in handling a mixed fleet of vehicles by generating solutions that exploit the EVs usage within their capacity and driving range constraints. The numerical experiments showed that the proposed approach is effective not only on HF-CVRP instances, but also on classical CVRP instances and that instances with thousands of customers and hundreds of vehicles can be handled in a reasonable time and can be more effective than other available implementations. However, from a performance point of view the ILS is better than the MCTS algorithm. Moreover, the experimental

results show that the algorithm can sometimes struggle with instances where customers are randomly distributed. This is because clustering randomly distributed customers can produce clusters with outliers, therefore increasing the final solution. Additionally, the clustering algorithm does not consider the presence of natural barriers (i.e. sea, mountains, etc.) between the customers, which can increase the delivery time in real-world scenarios. These problems can be solved by using different clustering algorithms or by applying an additional step that appropriately reduces the number of clusters' outliers.

There are several interesting paths for future research. From an energy consumption perspective, a more realistic approach that incorporates temperature, speed changes, gradient and load distribution could be investigated. Rather than assuming that energy consumption is a linear function of the travelled distance. This could be achieved by deriving an 'energy matrix' instead of a distance matrix. This additional pre-processing step could lead to a more precise estimation of the driving range. Both an analytical approach as the one presented in Goeke and Schneider (2015) and a Machine Learning approach as proposed by De Cauwer et al. (2017) offering different solution approaches for some of the mentioned limitations could be investigated. From an algorithmic perspective the MCTS could be modified by using heuristic approaches during the rollout that could guide the search towards the exploration of better solutions. Moreover, to better show the flexibility of the proposed approach, future work could solve instances with multiple depots and consider customers with time window for deliveries.

Acknowledgments

This work has been conducted within the Knowledge Transfer Partnership between the University of Surrey and Basemap Ltd.

Data availability statement

The data that support the findings of this study are openly available in Smartana.org's repository at <https://www.smartana.org>, see VRP Instances.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Innovate UK [partnership number 11418].

Notes on contributors

Carmen Barletta is a software engineer at Gearset, Cambridge, United Kingdom. She worked as research associate for the University of Surrey and Basemap. Her projects included topics such as vehicle routing, Homomorphic Encryption and Convolutional Neural Networks. She received her MSc in Computer Science and Engineering from Politecnico di Milano, Italy.

Wolfgang Garn is an Associate Professor at the University of Surrey, United Kingdom. He was Programme director of Business Analytics and acting Head of the Business Transformation Department. His general research interests are in the area of Applied Mathematics, Business Analytics and Artificial Intelligence. He holds an MSc in Technical Mathematics and Computer Science and was awarded his PhD from Vienna University of Technology.

Christopher Turner is a Lecturer in Business Analytics at Surrey Business School, University of Surrey. He is research active in the fields of analytics, digital manufacturing, manufacturing simulation and human centric manufacturing. With his involvement in the successful completion of several UK research council funded projects he is experienced in the management of commercially focused applied projects. He has published over 90 papers in peer reviewed international journals and conferences. He is also a member of the IEEE task force on process mining.

Saber Fallah is an Associate Professor at the University of Surrey, United Kingdom. He is the director of Connected and Autonomous Vehicles Lab (CAV-Lab) within the department of Mechanical Engineering Sciences. His research interests include the areas of reinforced deep learning and its application on connected autonomous vehicles. His research has produced several patents.

ORCID

Carmen Barletta  <https://orcid.org/0000-0003-3954-9802>

Wolfgang Garn  <https://orcid.org/0000-0003-2278-8997>

Christopher Turner  <https://orcid.org/0000-0003-2812-3312>

Saber Fallah  <https://orcid.org/0000-0002-1298-1040>

References

- Abdo, A., Edelkamp, S., & Lawo, M. (2016). Nested rollout policy adaptation for optimizing vehicle selection in complex VRPs. In *IEEE 41st conference on local computer networks workshops* (pp. 213–221). IEEE.
- Amjadian, A., & Gharaei, A. (2021). An integrated reliable five-level closed-loop supply chain with multi-stage products under quality control and green policies: Generalised outer approximation with exact penalty. *International Journal of Systems Science: Operations & Logistics*, 1–21. <https://doi.org/10.1080/23302674.2021.1919336>
- Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2011). *The traveling salesman problem: A computational study*, Princeton Series in Applied Mathematics, Princeton University Press.

- Arneson, B., Hayward, R. B., & Henderson, P. (2010). Monte Carlo Tree search in Hex. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4), 251–258. <https://doi.org/10.1109/TCIAIG.2010.2067212>
- Askari, R., Sebt, M. V., & Amjadian, A. (2021). A multi-product EPQ model for defective production and inspection with single machine, and operational constraints: Stochastic programming approach. *Communications in Computer and Information Science*, 1458, 161–193. <https://doi.org/10.1007/978-3-030-89743-7>
- Baldacci, R., Toth, P., & Vigo, D. (2010, February). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1), 213–245. <https://doi.org/10.1007/s10479-009-0650-0>
- Beresneva, E., & Avdoshin, S. (2018). Analysis of mathematical formulations of capacitated vehicle routing problem and methods for their solution. In *Proceedings of the institute for system programming* (Vol. 30, pp. 233–250). Труды Института системного программирования РАН [Proceedings of the Institute for System Programming of the Russian Academy of Sciences]
- Brandão, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research*, 38(1), 140–151. <https://doi.org/10.1016/j.cor.2010.04.008>
- Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A survey of Monte Carlo Tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1), 1–43. <https://doi.org/10.1109/TCIAIG.2012.2186810>
- Cazenave, T. (2009). Nested Monte-Carlo search. In *Proceedings of the 21st international joint conference on artificial intelligence* (pp. 456–461). Morgan Kaufmann Publishers Inc.
- Cazenave, T., Lucas, J. Y., Kim, H., & Triboulet, T. (2020). Monte Carlo vehicle routing. In *ATT at ECAI 2020*.
- Chaslot, G. M. B., Winands, M. H., & Herik, H. (2008). Parallel Monte-Carlo Tree search. In *International conference on computers and games* (pp. 60–71). Springer.
- Choi, E., & Tcha, D. W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7), 2080–2095. <https://doi.org/10.1016/j.cor.2005.08.002>
- Comert, S. E., Yazgan, H. R., Kir, S., & Yener, F. (2018). A cluster first-route second approach for a capacitated vehicle routing problem: A case study. *International Journal of Procurement Management*, 11(4), 399–419. <https://doi.org/10.1504/IJPM.2018.092766>
- Coulom, R. (2009). The Monte-Carlo revolution in Go. In *The Japanese-French frontiers of science symposium (JFFoS 2008)* (Vol. 115).
- Croes, G. A. (1958). A method for solving Traveling-Salesman problems. *Operations Research*, 6(6), 791–812. <https://doi.org/10.1287/opre.6.6.791>
- Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 5(2), 266–288. <https://doi.org/10.1287/opre.5.2.266>
- Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1), 80–91. <https://doi.org/10.1287/mnsc.6.1.80>
- De Cauwer, C., Verbeke, W., Coosemans, T., Faid, S., & Van Mierlo, J. (2017). A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions. *Energies*, 10(5), 608. <https://doi.org/10.3390/en10050608>
- De Jaegere, N., Defraeye, M., & Van Nieuwenhuyse, I. (2014). *The vehicle routing problem: State of the art classification and review* (FEB Research Report KBI_1415).
- Edelkamp, S., Gath, M., Greulich, C., Humann, M., Herzog, O., & Lawo, M. (2016). Monte-Carlo Tree search for logistics. In *Commercial transport* (pp. 427–440). Springer.
- Erdogan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1), 100–114. <https://doi.org/10.1016/j.tre.2011.08.001>
- European Environment Agency (2016). Electric vehicles and the energy sector – impacts on europe's future emissions. Retrieved May 16, 2022, <https://www.eea.europa.eu/publications/electric-vehicles-and-the-energy/>
- Fern, A., & Lewis, P. (2011, March). Ensemble Monte-Carlo planning: An empirical study. In *Twenty-First International Conference on Automated Planning and Scheduling*.
- Garn, W. (2020). Closed form distance formula for the balanced multiple travelling salesmen. ArXiv preprint arXiv:2001.07749.
- Garn, W. (2021). Balanced dynamic multiple travelling salesmen: Algorithms and continuous approximations. *Computers & Operations Research*, 136, Article 105509. <https://doi.org/10.1016/j.cor.2021.105509>
- Geetha, S., Poonthalir, G., & Vanathi, P. (2009). Improved K-Means algorithm for capacitated clustering problem. *INFO-COMP Journal of Computer Science*, 8(4), 52–59.
- Gharaei, A., Amjadian, A., & Shavandi, A. (2021). An integrated reliable four-level supply chain with multi-stage products under shortage and stochastic constraints. *International Journal of Systems Science: Operations & Logistics*, 1–22. <https://doi.org/10.1080/23302674.2021.1958023>
- Gharaei, A., Hoseini Shekarabi, S. A., & Karimi, M. (2021). Optimal lot-sizing of an integrated EPQ model with partial backorders and re-workable products: An outer approximation. *International Journal of Systems Science: Operations & Logistics*, 1–17. <https://doi.org/10.1080/23302674.2021.2015007>
- Gharaei, A., Hoseini Shekarabi, S. A., Karimi, M., Pourjavad, E., & Amjadian, A. (2021). An integrated stochastic EPQ model under quality and green policies: Generalised cross decomposition under the separability approach. *International Journal of Systems Science: Operations & Logistics*, 8(2), 119–131. <https://doi.org/10.1080/23302674.2019.1656296>
- Gharaei, A., Karimi, M., & Hoseini Shekarabi, S. A. (2021). Vendor-managed inventory for joint replenishment planning in the integrated qualitative supply chains: Generalised benders decomposition under separability approach. *International Journal of Systems Science: Operations & Logistics*, 1–15. <https://doi.org/10.1080/23302674.2021.1962428>
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), 533–549. [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1)

- Goeke, D., & Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1), 81–99. <https://doi.org/10.1016/j.ejor.2015.01.049>
- Hashimoto, J., Kishimoto, A., Yoshizoe, K., & Ikeda, K. (2012). Accelerated UCT and its application to two-player games. In H. J. van den Herik and A. Plaat (Eds.), *Advances in computer games* (pp. 1–12). Springer Berlin Heidelberg.
- Hiermann, G., Hartl, R. F., Puchinger, J., & Vidal, T. (2019). Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, 272(1), 235–248. <https://doi.org/10.1016/j.ejor.2018.06.025>
- Justesen, N., Mahlmann, T., Risi, S., & Togelius, J. (2017). Playing multiaction adversarial games: Online evolutionary planning versus tree search. *IEEE Transactions on Games*, 10(3), 281–291. <https://doi.org/10.1109/TCIAIG.2017.2738156>
- Kartal, B., Nunes, E., Godoy, J., & Gini, M. (2016, July). Monte Carlo Tree search with branch and bound for multi-robot task allocation.
- Kizilates, G., & Nuriyeva, F. (2013). On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in computational science, engineering and information technology* (pp. 111–118). Springer International Publishing.
- Kolmogorov, V. (2009). Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1), 43–67. <https://doi.org/10.1007/s12532-009-0002-8>
- Lai, D. S., Demirag, O. C., & Leung, J. M. (2016). A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review*, 86, 32–52. <https://doi.org/10.1016/j.tre.2015.12.001>
- Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5), 1050–1073. <https://doi.org/10.1287/opre.33.5.1050>
- Lebeau, P., De Cauwer, C., Van Mierlo, J., Macharis, C., Verbeke, W., & Coosemans, T. (2015, January). Conventional, hybrid, or electric vehicles: Which technology for an urban distribution centre?. *The Scientific World Journal*, 2015. <https://doi.org/10.1155/2015/302867>
- Li, F., Golden, B., & Wasil, E. (2007). A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(9), 2734–2742. <https://doi.org/10.1016/j.cor.2005.10.015>
- Lubosch, M., Kunath, M., & Winkler, H. (2018). Industrial scheduling with Monte Carlo Tree search and machine learning. *Procedia CIRP*, 72, 1283–1287. <https://doi.org/10.1016/j.procir.2018.03.171>
- Malinen, M. I., & Fränti, P. (2014). Balanced K-means for clustering. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 32–41). Springer.
- Mańdziuk, J. (2010). *Knowledge-free and learning-based methods in intelligent game playing* (Vol. 276).
- Mańdziuk, J. (2011). Towards cognitively plausible game playing systems. *IEEE Computational Intelligence Magazine*, 6(2), 38–51. <https://doi.org/10.1109/MCI.2011.940626>
- Mańdziuk, J., & Nejman, C. (2015, June). UCT-based approach to capacitated vehicle routing problem. In *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)* (Vol. 9120, pp. 679–690). Springer.
- Meliani, Y., Hani, Y., Elhaq, S. L., & El Mhamedi, A. (2019). A developed Tabu search algorithm for heterogeneous fleet vehicle routing problem. *IFAC-PapersOnLine*, 52(13), 1051–1056. <https://doi.org/10.1016/j.ifacol.2019.11.334>
- Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2), 201–232. <https://doi.org/10.1007/s10732-011-9186-y>
- Perron, L., & Furnon, V. (2019). *OR-Tools*. Retrieved May 16, 2022, <https://developers.google.com/optimization/>.
- Sbai, I., & Krichen, S. (2022, March). Vehicle routing problems with loading constraints: An overview of variants and solution methods. In *Optimization and machine learning* (pp. 1–23). John Wiley & Sons.
- Schneider, M., Stenger, A., & Goeke, D. (2014, March). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4), 500–520. <https://doi.org/10.1287/trsc.2013.0490>
- Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016, January). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484–489. <https://doi.org/10.1038/nature16961>
- Soejima, Y., Kishimoto, A., & Watanabe, O. (2011, January). Evaluating root parallelization in Go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4), 278–287. <https://doi.org/10.1109/TCIAIG.2010.2096427>
- Subramanian, A., Penna, P. H. V., Uchoa, E., & Ochi, L. S. (2012). A hybrid algorithm for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 221(2), 285–295. <https://doi.org/10.1016/j.ejor.2012.03.016>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. A Bradford Book.
- Taleizadeh, A. A., Safaei, A. Z., Bhattacharya, A., & Amjadian, A. (2022, March). Online peer-to-peer lending platform and supply chain finance decisions and strategies. *Annals of Operations Research*, 1–31. <https://doi.org/10.1007/s10479-022-04648-w>
- Tanabe, Y., Yoshizoe, K., & Imai, H. (2009). A study on security evaluation methodology for image-based biometrics authentication systems. In *2009 IEEE 3rd international conference on biometrics: Theory, applications, and systems* (pp. 1–6). IEEE.
- Tong, B. K. B., Ma, C. M., & Sung, C. W. (2011). A Monte-Carlo approach for the endgame of Ms. Pac-Man. In *IEEE conference on computational intelligence and games* (pp. 9–15). IEEE.
- Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., & Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3), 845–858. <https://doi.org/10.1016/j.ejor.2016.08.012>
- United States Environmental Protection Agency (2020). Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990–2018. Retrieved May 16, 2022, <https://www.epa.gov/ghgemissions/inventory-us-greenhouse-gas-emissions-and-sinks-1990-2018>.

- Venables, W. N., & Ripley, B. D. (2013). *Modern applied statistics with s-plus*. Springer Science & Business Media.
- Vidal, T. (2022, April). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140, Article 105643. <https://doi.org/10.1016/j.cor.2021.105643>
- Yuan, Y., Cattaruzza, D., Ogier, M., & Semet, F. (2020). A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for routing problems with time windows. *Operations Research Letters*, 48(2), 167–169. <https://doi.org/10.1016/j.orl.2020.01.008>
- Zhou, Y., Grygorash, O., & Hain, T. F. (2011). Clustering with minimum spanning trees. *International Journal on Artificial Intelligence Tools*, 20(01), 139–177. <https://doi.org/10.1142/S0218213011000061>
- Zhuang, J., Chu, S. C., Hu, C. C., Liao, L., & Pan, J. S. (2022, March). Advanced phasmatodea population evolution algorithm for capacitated vehicle routing problem. *Journal of Advanced Transportation*, 2022, 1–20. <https://doi.org/10.1155/2022/9241112>

Appendices

Appendix 1. CVRP Experimental results

Benchmark test-instances from Uchoa et al. (2017) for the CVRP were used to compare the best-known solution (BKS) to the here introduced algorithms. The solutions and relevant measures are shown in Tables A1 and A2. Below is a summary of the notation used for the instances, which was reproduced partly from Uchoa et al. (2017). A more detailed explanation of the test-instances can be found in Uchoa et al. (2017) with two additional solution methods: Branch-Cut-and-Price (BCP) and unified hybrid genetic search (UHGS). Uchoa et al. (2017) also include the corresponding runtime observations. The runtime for the Google OR and the MCTS algorithm were capped at 30 and 25 minutes, respectively. n represents the number of customers in the table below. The number of vehicles is k . The depot positioning (DP) can be: Central (C), i.e. depot is in the

centre of the grid, point (50 0,50 0); Eccentric (E), i.e. depot is in the corner of the grid, point (0,0); and Random (R), i.e. depot is in a random point of the grid. The customer positioning (CP) can be:

- Random (R) – all customers are positioned in random points of the grid;
- Clustered (C) – a number S of customers that act as cluster seeds is picked from an uniform discrete distribution $U[3,8]$. The S seeds are randomly positioned in the grid, and attract with an exponential decay;
- Random-clustered (RC) – half of the customers are clustered and the remaining ones are randomly positioned.

There are several demand distributions (DD):

- Unitary (U) – all demands have value 1;
- Small values, large CV (1–10) – demands from $U[1,10]$;
- Small values, small CV (5–10) – demands from $U[5,10]$;
- Large values, large CV (1–100) – demands from $U[1,100]$;
- Large values, small CV (50–100) – demands from $U[50,100]$;
- Depending on quadrant (Q);
- Many small values, few large values (SL).

The best-known solution distance is abbreviated with BKS. The number of used vehicles by the BKS is abbreviated with NV. The distances are captured are d_i , d_g and d_m representing the solution values obtained from the iterated local search (ILS), Google OR algorithm and the Monte–Carlo–Tabu–Search heuristic respectively. Please note that the ILS results originate from Uchoa et al. (2017) and the interested reader is referred to their work for further details. The distance ratio, and routed ratio are abbreviated with RD and RR (see details in Section 5).

Appendix 2. HF-CVRP Experimental results

Table A3 shows test-instances adjusted for the Hybrid Fleet – CVRP.

Table A1. Test-instances (1–50) (a) comparing BKS to ILS, GOR and MCTS distances; (b) GOR and MCTS distance ratios; (c) and MCTS routed parcels ratio.

#	Name	n	DP	CP	DD	BKS	NV	d_i	d_g	d_m	RD_g	RD_m	RR_m
1	X-n101-k25	100	R	RC (7)	1–100	27,591	26	27,591	29,768	36,965	1.08	1.34	1.00
2	X-n106-k14	105	E	C (3)	50–100	26,362	14	26,376	27,046	28,682	1.03	1.09	1.00
3	X-n110-k13	109	C	R	5–10	14,971	13	14,971	15,623	20,200	1.04	1.35	1.00
4	X-n115-k10	114	C	R	SL	12,747	10	12,747	13,420	20,236	1.05	1.59	1.00
5	X-n120-k6	119	E	RC (8)	U	13,332	6	13,338	14,251	15,307	1.07	1.15	1.00
6	X-n125-k30	124	R	C (5)	Q	55,539	30	55,674		64,107		1.15	0.99
7	X-n129-k18	128	E	RC (8)	1–10	28,940	18	28,998	31,250	37,606	1.08	1.30	1.00
8	X-n134-k13	133	R	C (4)	Q	10,916	13	10,947	13,627	15,370	1.25	1.41	1.00
9	X-n139-k10	138	C	R	5–10	13,590	10	13,603	14,877	17,350	1.09	1.28	1.00
10	X-n143-k7	142	E	R	1–100	15,700	7	15,745	17,376	18,643	1.11	1.19	1.00
11	X-n148-k46	147	R	RC (7)	1–10	43,448	47	43,452		72,298		1.66	0.99
12	X-n153-k22	152	C	C (3)	SL	21,220	23	21,400		36,059		1.70	0.99
13	X-n157-k13	156	R	C (3)	U	16,876	13	16,876	17,455	18,701	1.03	1.11	1.00
14	X-n162-k11	161	C	RC (8)	50–100	14,138	11	14,160	14,911	17,949	1.05	1.27	1.00
15	X-n167-k10	166	E	R	5–10	20,557	10	20,609	22,745	24,802	1.11	1.21	1.00
16	X-n172-k51	171	C	RC (5)	Q	45,607	53	45,616		80,082		1.76	0.98
17	X-n176-k26	175	E	R	SL	47,812	26	48,250		57,406		1.20	0.99
18	X-n181-k23	180	R	C (6)	U	25,569	23	25,572	26,513	28,823	1.04	1.13	1.00
19	X-n186-k15	185	R	R	50–100	24,145	15	24,186	26,050	32,376	1.08	1.34	1.00
20	X-n190-k8	189	E	C (3)	1–10	16,980	8	17,143	18,180	19,539	1.07	1.15	1.00
21	X-n195-k51	194	C	RC (5)	1–100	44,225	53	44,234		78,090		1.77	0.97
22	X-n200-k36	199	R	C (8)	Q	58,578	36	58,697		76,443		1.30	0.96
23	X-n204-k19	203	C	RC (6)	50–100	19,565	19	19,625	21,587	25,846	1.10	1.32	1.00
24	X-n209-k16	208	E	R	5–10	30,656	16	30,765	32,508	36,545	1.06	1.19	1.00
25	X-n214-k11	213	C	C (4)	1–100	10,856	11	11,127	12,880	14,285	1.19	1.32	0.98
26	X-n219-k73	218	E	R	U	117,595	73	117,595	118,409	161,497	1.01	1.37	1.00
27	X-n223-k34	222	R	RC (5)	1–10	40,437	34	40,534	44,611	71,482	1.10	1.77	0.99
28	X-n228-k23	227	R	C (8)	SL	25,742	23	25,796	30,596	49,243	1.19	1.91	1.00
29	X-n233-k16	232	C	RC (7)	Q	19,230	17	19,337		28,115		1.46	0.99
30	X-n237-k14	236	E	R	U	27,042	14	27,079	29,009	32,127	1.07	1.19	1.00
31	X-n242-k48	241	E	R	1–10	82,751	48	82,874	86,573	116,846	1.05	1.41	0.98
32	X-n247-k50	246	C	C (4)	SL	37,274	51	37,507		59,371		1.59	0.98
33	X-n251-k28	250	R	RC (3)	5–10	38,684	28	38,840	40,472	49,758	1.05	1.29	1.00
34	X-n256-k16	255	C	C (8)	50–100	18,880	17	18,884		24,181		1.28	0.95
35	X-n261-k13	260	E	R	1–100	26,558	13	26,869	29,415	37,462	1.11	1.41	1.00
36	X-n266-k58	265	R	RC (6)	5–10	75,478	58	75,563		114,947		1.52	0.96
37	X-n270-k35	269	C	RC (5)	50–100	35,291	36	35,363		64,279		1.82	0.97
38	X-n275-k28	274	R	C (3)	U	21,245	28	21,256	22,491	26,826	1.06	1.26	1.00
39	X-n280-k17	279	E	R	SL	33,503	17	33,769	38,220	45,426	1.14	1.36	1.00
40	X-n284-k15	283	R	C (8)	1–10	20,226	15	20,449	22,545	26,503	1.11	1.31	1.00
41	X-n289-k60	288	E	RC (7)	Q	95,151	61	95,451		144,263		1.52	0.96
42	X-n294-k50	293	C	R	1–100	47,167	51	47,255		96,139		2.04	0.97
43	X-n298-k31	297	R	R	1–10	34,231	31	34,356	40,336	71,040	1.18	2.08	0.99
44	X-n303-k21	302	C	C (8)	1–100	21,744	21	21,896	23,489	36,978	1.08	1.70	1.00
45	X-n308-k13	307	E	RC (6)	SL	25,859	13	26,101	30,500	41,001	1.18	1.59	1.00
46	X-n313-k71	312	R	RC (3)	Q	94,044	72	94,297		146,160		1.55	0.95
47	X-n317-k53	316	E	C (4)	U	78,355	53	78,356	79,224	92,208	1.01	1.18	1.00
48	X-n322-k28	321	C	R	50–100	29,866	28	29,991		39,132		1.31	0.98
49	X-n327-k20	326	R	RC (7)	5–10	27,556	20	27,812	29,626	37,545	1.08	1.36	1.00
50	X-n331-k15	330	E	R	U	31,103	15	31,236	33,579	39,820	1.08	1.28	1.00

Table A2. Test-instances (51–100) (a) comparing BKS to ILS, GOR and MCTS distances; (b) GOR and MCTS distance ratios; (c) and MCTS routed parcels ratio.

#	Name	n	DP	CP	DD	BKS	NV	d_i	d_g	d_m	RD_g	RD_m	RR_m
51	X-n336-k84	335	E	R	Q	139,197	86	139,461		205,000		1.47	0.96
52	X-n344-k43	343	C	RC (7)	5–10	42,099	43	42,284		68,251		1.62	0.94
53	X-n351-k40	350	C	C (3)	1–100	25,946	41	26,150		55,564		2.14	0.98
54	X-n359-k29	358	E	RC (7)	1–10	51,509	29	52,077	55,413	81,263	1.08	1.58	0.99
55	X-n367-k17	366	R	C (4)	SL	22,814	17	23,003	25,294	42,113	1.11	1.85	1.00
56	X-n376-k94	375	E	R	U	147,713	94	147,713	148,791	187,188	1.01	1.27	1.00
57	X-n384-k52	383	R	R	50–100	66,081	53	66,373		112,516		1.70	0.97
58	X-n393-k38	392	C	RC (5)	5–10	38,269	38	38,457		73,938		1.93	0.98
59	X-n401-k29	400	E	C (6)	Q	66,243	29	66,715		87,239		1.32	0.99
60	X-n411-k19	410	R	C (5)	SL	19,718	19	19,955	22,913	49,105	1.16	2.49	1.00
61	X-n420-k130	419	C	RC (3)	1–10	107,798	130	107,838		196,491		1.82	0.97
62	X-n429-k61	428	R	R	50–100	65,501	62	65,747		118,458		1.81	0.96
63	X-n439-k37	438	C	RC (8)	U	36,391	37	36,442	38,442	50,204	1.06	1.38	1.00
64	X-n449-k29	448	E	R	1–100	55,358	29	56,205	62,426	87,534	1.13	1.58	0.99
65	X-n459-k26	458	C	C (4)	Q	24,181	26	24,462	27,150	40,000	1.12	1.65	0.98
66	X-n469-k138	468	E	R	50–100	221,909	140	222,182		327,268		1.47	0.95
67	X-n480-k70	479	R	C (8)	5–10	89,535	70	89,871		141,204		1.58	0.97
68	X-n491-k59	490	R	RC (6)	1–100	66,633	60	67,227	72,759	124,779	1.09	1.87	0.98
69	X-n502-k39	501	E	C (3)	U	69,253	39	69,347	70,302	77,886	1.02	1.12	0.93
70	X-n513-k21	512	C	RC (4)	1–10	24,201	21	24,434	27,631	35,306	1.14	1.46	1.00
71	X-n524-k153	523	R	R	SL	154,594	155	155,005		275,810		1.78	0.97
72	X-n536-k96	535	C	C (7)	Q	95,122	97	95,701		171,406		1.80	0.95
73	X-n548-k50	547	E	R	U	86,710	50	86,874	89,529	103,541	1.03	1.19	0.91
74	X-n561-k42	560	C	RC (7)	1–10	42,756	42	43,131	48,083	95,856	1.12	2.24	0.99
75	X-n573-k30	572	E	C (3)	SL	50,780	30	51,173	53,775	74,673	1.06	1.47	1.00
76	X-n586-k159	585	R	RC (4)	5–10	190,543	159	190,919		330,376		1.73	0.95
77	X-n599-k92	598	R	R	50–100	108,813	94	109,384		182,625		1.68	0.94
78	X-n613-k62	612	C	R	1–100	59,778	62	60,444	66,721	140,937	1.12	2.36	0.99
79	X-n627-k43	626	E	C (5)	5–10	62,366	43	62,906		99,502		1.60	0.96
80	X-n641-k35	640	E	RC (8)	50–100	63,839	35	64,606	67,679	91,134	1.06	1.43	0.98
81	X-n655-k131	654	C	C (4)	U	106,780	131	106,782	108,015	143,602	1.01	1.34	0.80
82	X-n670-k130	669	R	R	SL	146,705	134	147,676		282,127		1.92	0.96
83	X-n685-k75	684	C	RC (6)	Q	68,425	75	68,988	79,180	161,165	1.16	2.36	0.98
84	X-n701-k44	700	E	RC (7)	1–10	82,292	44	83,042	89,250	119,327	1.08	1.45	0.98
85	X-n716-k35	715	R	C (3)	1–100	43,525	35	44,172	47,155	69,093	1.08	1.59	0.99
86	X-n733-k159	732	C	R	1–10	136,366	160	137,045		272,757		2.00	0.95
87	X-n749-k98	748	R	C (8)	1–100	77,700	98	78,276	83,454	151,219	1.07	1.95	0.96
88	X-n766-k71	765	E	RC (7)	SL	114,683	71	115,738	127,232	211,649	1.11	1.85	0.99
89	X-n783-k48	782	R	R	Q	72,727	48	73,723	79,555	126,510	1.09	1.74	0.98
90	X-n801-k40	800	E	R	U	73,587	40	74,006	76,321	93,660	1.04	1.27	1.00
91	X-n819-k171	818	C	C (6)	50–100	158,611	173	159,425		287,437		1.81	0.93
92	X-n837-k142	836	R	RC (7)	5–10	194,266	142	195,027		299,429		1.54	0.94
93	X-n856-k95	855	C	RC (3)	U	89,060	95	89,278	91,616	126,775	1.03	1.42	0.89
94	X-n876-k59	875	E	C (5)	1–100	99,715	59	100,417	103,743	161,508	1.04	1.62	0.98
95	X-n895-k37	894	R	R	50–100	54,172	38	54,959		82,353		1.52	0.93
96	X-n916-k207	915	E	RC (6)	5–10	329,836	208	330,948		480,880		1.46	0.94
97	X-n936-k151	935	C	R	SL	133,105	159	134,530		342,352		2.57	0.96
98	X-n957-k87	956	R	RC (4)	U	85,672	87	85,937	88,972	119,278	1.04	1.39	0.91
99	X-n979-k58	978	E	C (6)	Q	119,194	58	120,253		170,613		1.43	0.97
100	X-n1001-k43	1000	R	R –	1–10	72,742	43	73,985	79,437	117,932	1.09	1.62	0.98

Table A3. Hybrid Fleet – CVRP test-instances with electric vehicles (EV) and vehicles with internal combustion engine (ICE).

Name	R	EVs	ICEs	RoutedRatio	DistanceRatio	NotUsedCars	EVUsage
X-n120-k6	2400	3	4	1.00	1.21	1	0.92
X-n204-k19	1030	10	11	0.98	1.54	0	0.88
X-n439-k37	983	19	20	0.93	1.37	0	0.87
X-n573-k30	1689	15	16	0.96	1.38	0	0.98
X-n801-k40	1832	20	22	0.91	1.27	0	0.90