Demand Selection for VRP with Emission Quota

Farid Najar, Dominique Barth, Yann Strozecki

DAVID LAB, UVSQ, Université Paris-Saclay
{first-name.last-name}@uvsq.fr

Abstract. Combinatorial optimization (CO) problems are traditionally addressed using Operations Research (OR) methods, including metaheuristics. In this study, we introduce a demand selection problem for the Vehicle Routing Problem (VRP) with an emission quota, referred to as QVRP. The objective is to minimize the number of omitted deliveries while respecting the pollution quota. We focus on the demand selection part, called Maximum Feasible Vehicle Assignment (MFVA), while the construction of a routing for the VRP instance is solved using classical OR methods. We propose several methods for selecting the packages to omit, both from machine learning (ML) and OR. Our results show that, in this static problem setting, classical OR-based methods consistently outperform ML-based approaches.

Keywords: Demand selection · Reinforcement learning · Operations research.

1 Introduction

Due to climate change, nations must implement measures to reduce emissions and mitigate its critical and far-reaching consequences. Local authorities, in particular, are working to minimize their environmental footprint by restricting the emission of greenhouse gases such as CO_2 and other pollutants. One example is the establishment of Low Emission Zones, developed in several European cities to limit access for polluting delivery vehicles. This study focuses on supply chain management, with an emphasis on reducing emissions generated during the delivery process.
Under stringent emission constraints or quotas, transporters are compelled to delay or cancel certain demands. The primary objective of this work is to determine the optimal selection of demands to maximize the number of accepted deliveries.
We decompose the **demand selection** problem into two distinct layers, representing two simpler subproblems where the ML methods can be used to select demands. The first layer, referred to as the **assignment layer**, selects the demands to be delivered and, optionally, assigns a vehicle to each demand. The second layer, termed the **routing layer**, computes the routes for the resulting VRP instance based on the vehicles' cost matrices. For the routing layer, we employ classical methods, such as the guided local search (GLS) algorithm from the OR-Tools library [27].
Subsequently, we employ well-established methods, such as dynamic programming and metaheuristics, to establish baseline results. Building on this foundation, we focus on

learning-based approaches to address the problem. Our objective is to design and evaluate methods that combine operational research (OR) techniques and learning-based strategies, incorporating varying levels of hybridization with the routing component, which is solved using OR techniques. We compare the performance of these methods in terms of both computational efficiency and solution quality.

Related Works Traditionally, combinatorial optimization problems have been addressed using operations research (OR) methods such as linear programming (LP), dynamic programming (DP), and metaheuristics [12]. In the context of our study, some researchers have sought to reduce emissions in the VRP alongside costs by employing integer linear programming within multi-objective optimization frameworks [25]. While LP and DP encounter scalability challenges for large problem instances, metaheuristic methods such as simulated annealing (SA) often provide robust performance even in these cases [16].

The environmental impact of VRPs has been the focus of several studies [11], most of which rely predominantly on OR techniques, particularly metaheuristics. For example, [37] incorporates fuel costs, carbon emissions, and vehicle usage costs into the traditional VRP, creating a low-carbon routing problem model. Similarly, [29] integrates customer satisfaction alongside cost and carbon emissions, revealing a trade-off between reducing emissions and maintaining customer satisfaction.

Other studies approach this problem as a multi-objective optimization task, where carbon emissions are optimized alongside criteria such as cost and waiting time [9,28,19,4].

Research has explored the application of reinforcement learning (RL) across various domains related to OR, including combinatorial optimization [24], scheduling [7], and specific supply chain challenges [6,8,26]. In [21], a graph convolutional network (GCN) is employed to predict high-granularity, time-dependent traffic speeds, combined with a hybrid genetic algorithm and adaptive variable neighborhood search to achieve low-carbon vehicle routing.

Several studies have integrated machine learning (ML) and deep learning into metaheuristics to enhance their performance [35,32]. Others have employed specialized neural network architectures to tackle combinatorial optimization problems [5,22], while multi-agent approaches have also gained attention [3,13,23].

For solving the Traveling Salesman Problem (TSP) and Vehicle Routing Problems (VRPs), traditional methods remain highly effective. The state-of-the-art solver *Concorde* [1] provides optimal solutions for TSP, the Lin-Kernighan Heuristic (LKH) [14] is an approximate algorithm capable of solving both TSP and VRP, often yielding near-optimal solutions, and Hybrid Genetic Search (HGS) [36] has proven to be one of the best metaheuristic algorithms for VRP. However, both heuristic and metaheuristic methods lack formal asymptotic guarantees on performance.

In recent years, learning-based approaches have been applied to VRPs, particularly using deep neural networks (DNNs). Some studies leverage DNNs to learn heuristics for dynamic programming [17], while others utilize RL or multi-agent RL approaches [26,13,23], or integrate attention mechanisms [18]. Although these methods have shown interesting results, they generally fail to outperform classical OR methods in terms of efficiency and solution quality.

A promising direction lies in hybrid approaches, where an agent learns to partition the problem and delegates tasks to local searchers based on metaheuristics [20]. This method has demonstrated strong results, outperforming other learning-based approaches in scalability, computational efficiency, and solution quality when compared to traditional OR techniques.

Our Contributions We introduce **MFVA** a demand selection problem with a static set of demands, formulated as a combinatorial optimization problem, which we decompose into an assignment layer and a routing layer. To the best of our knowledge, this problem is rarely addressed in the literature on delivery and commodity transportation services, as classical VRPs often assume that all deliveries are to be served, or it is outsourced by paying a penalty [33].
We propose and implement several approaches to solve this problem, including operations research (OR) methods and reinforcement learning (RL)-based techniques. In particular, we try the state of the art actor-critic method Proximal Policy Iteration [31] to train a neural network to solve the selection problem, and we also try decentralized multi-agent learning methods such as EXP3 [2] and LRI [30]. Our findings demonstrate that classical OR methods, particularly those based on metaheuristics, remain more efficient and outperform RL-based methods in this context.

2 Problem Setting

We denote the set of integers $\{1, \dots, n\}$ by $[n]$. We model a transporter that seeks to distribute packages to d different destinations from one single hub or warehouse. We represent the destinations (or equivalently the packages) by integers in $[d]$, and the hub vertex is represented by 0. We let q be the function from $[d]$ to $\mathbb{N}^*$, such that $q(i)$ represents the quantity of product to be delivered at destination i. We are given a distance matrix D between all pairs of destinations, we denote by $D[i, j]$ the distance between i and j.
A **route** r is an ordered list of destinations in $[d]$, without repetition, which begins and finishes with the hub vertex 0: $(d_0 = 0, d_1, \dots, d_k = 0)$. The **length** of a route r is the sum of the distances between consecutive destinations in r: $\mathrm{len}(r) = \sum_{i=0}^{k-1} D[d_i, d_{i+1}]$. We denote by $\mathcal{D}(r)$ the set of destinations reached in route R, that is all d_i for $0 < i < k$. We define the **load** of a route as the total quantity which must be delivered to destinations in the route: $\mathrm{load}(r) = \sum_{i=1}^{k-1} q(d_i)$.
A **vehicle** v is a triple (Cap_v, Ef_v, Cf_v), where Cap_v is an integer representing the **capacity** of the vehicle, Ef_v and Cf_v are rationals, respectively the **emission factor**, and the **cost factor** of the vehicle. We define the emission of a vehicle v following route r as $\mathrm{em}_v(r) = Ef_v \times \mathrm{len}(r)$ and the cost of a vehicle v following r by $\mathrm{cost}_v(r) = Cf_v \times \mathrm{len}(r)$. In this work, we have chosen to make emission and cost depend linearly on the length of the route, which is enough to model real logistic problems quite accurately. The different values of the coefficients Cf_v and Ef_v, enable us to model different types of vehicles and engine technologies.
The fleet of a transporter $\mathcal{V}$ is a set of vehicles. Each vehicle has its own capacity, emission factor and cost factor. A **routing** R is a function mapping each vehicle $v \in \mathcal{V}$

to a route R_v, such that *each vehicle transports less than or equal to its capacity*: $\mathrm{load}(R_v) \leq Cap_v$. We denote by $\mathcal{D}_0(R)$ the set of packages which are not delivered when following the routes of R: $\mathcal{D}_0(R) = \mathcal{D} \setminus \cup_{v \in \mathcal{V}} \mathcal{D}(R_v)$. We define the cost of a routing R as the sum of the costs of the vehicles: $\mathrm{cost}(R) = \sum_{v \in \mathcal{V}} \mathrm{cost}_v(R_v)$. Similarly, the emission of a routing R is the sum of the emissions of the vehicles: $\mathrm{em}(R) = \sum_{v \in \mathcal{V}} \mathrm{em}_v(R_v)$.

In our settings, the transporter must respect strong emission regulations set by governing bodies, as represented by an emission quota denoted by Q. A routing R is admissible only if its emissions are less than the quota: $\mathrm{em}(R) \leq Q$. Therefore, we tackle the **vehicle routing problem under a quota** denoted by **QVRP** defined as follows: given a matrix of distances D, a quantity function q, a set of vehicles $\mathcal{V}$, a quota Q, compute a routing R such that $\mathrm{em}(R) \leq Q$, which minimizes $\mathrm{cost}(R)$.

However, the particularity of our problem is that the constraint on emissions cannot be satisfied while delivering all packages: QVRP is typically infeasible. Hence, the transporter *has to omit some packages*, and we face the problem of selecting the packages to omit. Consequently, the routing R might not include all destinations. For quality of service reasons, our main objective in this paper is to minimize the undelivered quantity, that we define as $\mathrm{Oq}(R)$ for Omitted quantity: $\mathrm{Oq}(R) = \sum_{i \in \mathcal{D}_0(R)} q(i)$. Moreover, given an omitted quantity, if possible minimum, we want to minimize the cost of the routing used to deliver the non-omitted packages.

Thus, we introduce the problem **maximum feasible vehicle assignment (MFVA)** defined as follows : given a matrix of distances D, a quantity function q, a set of vehicles $\mathcal{V}$, a quota Q, compute a vehicle assignment with maximum cardinality for which there exists a (minimum cost) routing R such that $\mathrm{em}(R) \leq Q$, which minimizes in lexicographic order $(\mathrm{Oq}(R), \mathrm{cost}(R))$.

A **vehicle assignment** a is a mapping from $[d]$ to $\{0\} \cup \mathcal{V}$. If $a(i) = v$, it means that vehicle v delivers the package at destination i, while $a(i) = 0$ means that no vehicle delivers the package at destination i. A routing R respects a vehicle assignment a if for each vehicle v, $\mathcal{D}(R_v) = a^{-1}(v)$. The cardinal of an assignment a is the number of $i \in [d]$ such that $a(i) \in \mathcal{V}$. We alternatively consider the simpler **omission assignment** a, a function from $[d]$ to $\{0, 1\}$, where $a(i) = 0$ means that no vehicle delivers the package at i, while $a(i) = 1$ means that a vehicle must deliver it. A routing R respects an omission assignment a if $\mathcal{D}_0(R) = a^{-1}(0)$.

To make MFVA more tractable, we decompose it into two interacting subproblem layers: (i) choosing an assignment and (ii) finding the best routing respecting this assignment.

Assignment Layer. This layer consists in choosing a vehicle assignment which is then given to the routing layer, which produces a routing respecting this assignment. The choice of a vehicle assignment (and the corresponding routing layer resolution) is repeated many times, by using different algorithms, to converge to an efficient assignment.

We model the routing layer by a function f which returns a routing from an instance $I = (D, q, \mathcal{V}, Q)$ of MFVA and an assignment a. The computed routing $f(I, a)$ must satisfy the capacity constraints given in $\mathcal{V}$ and must respect the vehicle assignment a

(or the omission assignment a), but it may not respect the emission quota Q if it is not possible.
Hence, the problem solved in the assignment layer is the following, given I and f, compute a such that $\mathrm{em}(f(I,a)) \leq Q$ and $(\mathrm{Oq}(f(I,a)), \mathrm{cost}(f(I,a)))$ is minimal in lexicographic order.

Routing Layer. In the routing layer, we are given a MFVA instance along with a vehicle/omission assignment, and we must produce a routing which respects the assignment and the quota if possible.
To solve MFVA on the assignment layer, an algorithm may query the routing layer many times, which is very time-consuming, since it corresponds to solving many instances of an NP-hard problem in the routing layer. Hence, we also consider a simpler variant of MFVA, where we are given a single routing R from the instance I, and we must remove packages *in the routing* R, to satisfy the emission constraint.
We say that R' is a **subrouting of** R, if it can be obtained by removing elements in the routes of R except the hub. We define the problem SHORTCUT as follows, given a set of vehicles $\mathcal{V}$, a routing R, a quota Q, an integer k, compute, if it exists, a subrouting R' such that $\mathrm{em}(R') \leq Q$ and $\mathrm{Oq}(R') \leq k$, which minimizes $\mathrm{cost}(R')$. This problem serves as a low complexity variant of MFVA to test the quality of our methods and to get bounds on the optimal solution. However, the optimal solution to this problem depends on the choice of the initial routing R and can be arbitrarily far from the solution of the original QVRP instance, as we prove in appendix A.

3 Solving the Routing Layer

Vehicle assignment When the routing layer is provided with a vehicle assignment, we solve a Capacitated Traveling Salesman Problem (CTSP) for each vehicle. The given vehicle assignment a determines the set of packages $a^{-1}(v)$ that each vehicle v must deliver. Since both emissions and costs are proportional to the total route length for a vehicle, minimizing the route length suffices to achieve the lowest emissions and costs while respecting a. We use the **Nearest Neighbor algorithm**, a simple heuristic, due to the need for solving this problem multiple times efficiently. This algorithm greedily constructs a route starting from the hub by selecting the cheapest unvisited destination from the current location. If adding a destination exceeds the vehicle's capacity, it is skipped.

Omission assignment When the routing layer is provided with an omission assignment, the problem is a QVRP, since no omissions on the selected packages. The primary challenge lies in balancing the dual objectives of cost minimization and quota adherence, which may conflict. To manage this, we combine cost and emissions into a single optimization value.
For each vehicle v, we define a new distance matrix between destinations $D_v^\lambda = Cf_vD + \lambda Ef_vD$ where λ is a constant. The parameter λ[1] must be large enough to prioritize

[1] The value of λ chosen for our experiments is given in appendix C.

lower emissions, ensuring compliance with the quota, but not so large that it disregards cost optimization.
To solve QVRP, we use the classical VRP approach with the distance matrix D_v^λ for each vehicle v. This is implemented using the `or-tools` library, a widely used open-source tool for solving combinatorial optimization problems. For relatively small instances such as ours, it provides optimal or near-optimal solutions. An initial solution is generated using a greedy algorithm similar to the Nearest Neighbor approach, applied sequentially to each vehicle. This solution is then refined using *Guided Local Search (GLS)* [12] until a predefined time budget is reached.

4 Methods for Solving the Assignment Layer

First, we propose different heuristics that solve MFVA on the assignment layer, either for omission assignment or for vehicle assignment, and an exact algorithm for the SHORTCUT problem.
Our objective is then to understand the relevance of learning-based methods, in particular Reinforcement Learning (RL) and multi-agent learning algorithms. We aim to compare these sophisticated techniques with classical OR methods. As explained earlier, we tackle the MFVA used in the routing layer in two ways: with vehicle assignment or with omission assignment. To compute the assignments, we adopt two main strategies: either beginning with an initial assignment and iteratively adjusting it, or independently deciding the fate of each package in a decentralized way.
Recall that we want to minimize in lexicographic order $(\mathrm{Oq}(f(I,a)), \mathrm{cost}(f(I,a)))$. To reduce our problem to the minimization of a single value, we use the loss function $\mathcal{L}_P$:

$$\mathcal{L}_P(R) = \mathrm{Oq}(R) \cdot P + \mathrm{cost}(R).$$

We let P be twice the largest distance from the hub. Hence, minimizing $\mathcal{L}_P(f(I,a))$ or $(\mathrm{Oq}(f(I,a)), \mathrm{cost}(f(I,a)))$ in lexicographic order is the same.

Greedy Removal

We first design a simple greedy algorithm. We begin with a an omission assignment such that no packages are omitted, that is $[d] = a^{-1}(1)$. From that, we get a routing respecting a on the instance I: $R = f(I,a)$.
We now build routing by removing elements from the routes of R. Let R_v be some route of R, and its elements are $R_{v,0}, R_{v,1}, \ldots, R_{v,last_v}$. Note that, for all $v \in \mathcal{V}$, $R_{v,0} = R_{v,last_v} = 0$ is the element representing the hub where our vehicles are loaded and come back after delivery. In this algorithm, we fix the routing R, hence an assignment a can be interpreted as a set of pairs (v,j) representing the destination removed from R. We define the subrouting $R(a)$ of R as the routing R where $R_{v,j}$ is removed from R_v for all $(v,j) \in a$. We always have $j > 0$, since we cannot remove the first element, which represents the hub and not a package destination.
To guide our algorithm, we must take into account the value of the current omission assignment, but also bias it towards a routing which respects the quota. To do so, we

define a new objective function $g(a) = \mathcal{L}_P(a) + \lambda(\mathrm{em}(R(a)) - Q)^+$ where $\lambda > 0$ is the same emission penalty coefficient defined in Section 3. Our greedy algorithm, called **Greedy Removal** or `Greedy` for short, select at each step some pair (v, j) to remove from a (setting $a = a \setminus \{(v, j)\}$) which minimizes g. When we get an omission assignment a such that $R(a)$ respects the quota, we return it.
The described algorithm work on a fixed routing R, hence it solves the simple SHORTCUT problem.

Dynamic Programming

We give a dynamic program to optimally solve the SHORTCUT problem, the simplified version of the assignment layer where the routing is fixed and destinations are removed in the routing. Solving the SHORTCUT problem produces the best way to remove elements of bounded total quantity k from a given routing; we can then find the smallest k such that the emission is below the quota Q.
We solve SHORTCUT on a routing R and a vehicle fleet $\mathcal{V}$. We let $N = |\mathcal{V}|$ and $V = \{v_1, \ldots, v_N\}$, hence we have an arbitrary order on the vehicles and the routes. Furthermore, we define the value function $\mathrm{Val}(i, j, k)$ as the smallest length of a subrouting $R(a)$ such that $\mathrm{Oq}(R(a)) = k$ and for all $(m, n) \in a$, $m < i$ or $m = i$ and $n < j$. In other words, $\mathrm{Val}(i, j, k)$ is the length of the best subrouting of R, with omitted quantity less than k, which is obtained by removing elements in the first i routes and in the i-th route only before the j-th destination.
We denote by $\Delta(v, j, s)$ the distance avoided when we remove s consecutive destinations before $R_{v,j}$ in the route R_v. We have

$$\Delta(v, j, s) = -D[R_{v,j-s-1}, R_{v,j}] + \sum_{t=0}^{s} D[R_{v,j-t-1}, R_{v,j-t}]$$

We denote by $\mathrm{Oq}(v, j, s)$ the quantity we remove in vehicle v if we remove s consecutive destinations before $R_{v,j}$ in the route R_v. We have

$$\mathrm{Oq}(v, j, s) = \sum_{t=j-s-1}^{j-1} q(R_{v,t})$$

The function Val satisfies the following equations for all v, j, k:

- $\mathrm{Val}(v, j, 0) = \mathrm{len}(R)$: nothing is removed, we have the length of the routing R,
- $\mathrm{Val}(v_{i+1}, 1, k) = \mathrm{Val}(v_i, last_{v_i}, k)$: removing quantity k before the first package of the route $R_{v_{i+1}}$ is the same as removing quantity k before the end of the route R_{v_i} (we use the implicit ordering of the vehicles),
- $\mathrm{Val}(v, j, k) = \min_{s \leq \min(v, j-1)} \mathrm{Val}(v, j - 1 - s, k - \mathrm{Oq}(v, j, s)) - \Delta(v, j, s)$ for $j > 1$: we may remove any number $s < j$ of consecutive elements before $R_{v,j}$ and we keep the choice with the lowest emission.

These equations enable us to compute $\mathrm{Val}(v_N, last_{v_N}, k)$ by dynamic programming. Recall that d is the number of destinations, and we assume that a destination appears at

most once in the routes of R. The complexity of computing all relevant values of Δ is in $O(d\min(k,d))$, since s, the number of consecutive destination removed, is bounded by k and also by d. The dynamic program must compute $O(kd)$ distinct values of Val and for each it evaluates a minimum of at most d values, hence it is of complexity $O(kd^2)$. From the computed values of Val and k, it is easy to derive a subrouting R' with $\mathrm{Oq}(R') \leq k$ such that R' has the smallest possible length in time $O(kd^2)$.
Assume now that all vehicles have the same emission factors and the same cost factors, denoted by Ef and Cf. Then, minimizing the length of a subrouting minimizes the emission and the cost simultaneously. Hence, to solve SHORTCUT, we only have to find the smallest k, such that $Ef \cdot \mathrm{Val}(N, last_N, k) \leq Q$.
Now, let us consider we have t different types of vehicle. Let $R^1, \ldots, R^t$ be the routing induced by a routing R on these t homogenous groups of vehicles. For each R^i and $l \leq k$, we compute in polynomial time by dynamic programming $R^i(l)$ the subrouting of the smallest length satisfying $\mathrm{Oq}(R^i(l)) \leq l$. We then generate all possible tuples of integers $(k_1, k_2, \ldots, k_t)$ such that they sum to k. For each of these tuples, we obtain a subrouting $R^1(k_1), \ldots, R^t(k_t)$ of R. We test all k in increasing order, until there is a subrouting $R^1(k_1), \ldots, R^t(k_t)$ with emission less than Q, and we select the one of smallest cost. This algorithm solves the problem SHORTCUT in time $O(\binom{k}{t}kd^2)$, where k is the omitted quantity in the optimal solution, since there are $\binom{k}{t}$ tuples $(k_1, k_2, \ldots, k_t)$ of sum k. It implies the following theorem :

Theorem 1. *When restricted to instances where the fleet of vehicles has a fixed number of different coefficients Ef and Cf,* SHORTCUT $\in$ P.

In our settings we typically have $t \leq 4$, and k is at most of the order of d, since the quantity of each destination is a few units. Hence, the described algorithm is efficient in practice. However, it is not possible to give a polynomial time algorithm for t unbounded, since SHORTCUT is a NP-hard problem for general instances as proved in Section A.

Simulated Annealing

We propose to use Simulated Annealing [12], a well-known metaheuristic method, to find a global minimum in a combinatorial problem. It achieves this by evaluating and selecting neighboring solutions in a stochastic manner, guided by a temperature parameter that gradually decreases over time.
Let us describe the simulated annealing algorithm solving MFVA for Omission Assignment, that we call Omission Assignment Simulated Annealing or `OA-SA`. To evaluate the quality of an omission assignment a during the simulated annealing, we use the function g, described previously, applied to $f(I, a)$. We define the **neighborhood** of a as the set of assignments a' such that a and a' differs on a single package: $|\{i \in [d] \mid a(i) \neq a'(i)\}| = 1$. The neighborhood exploration is done like a classical SA algorithm. The temperature decreases exponentially with some given factor.
We also define a simulated annealing algorithm solving MFVA for Vehicle Assignment, that we call Vehicle Assignment Simulated Annealing or `VA-SA`. We use the same value function for the solutions, which are vehicle assignments. The neighborhood of a

vehicle assignment is the set of vehicle assignments at distance 1, i.e. only one index is changed. In this context, however, the neighborhood is larger by a factor of $|\mathcal{V}|$, since we can assign the changed destination to any vehicle instead of adding or removing it.

Reinforcement Learning

A way of learning to find optimal behavior in an environment is by training through exploration and exploitation. This approach in machine learning is called *reinforcement learning (RL)*. RL is concerned with how the interacting agent ought to take actions in an environment to maximize their expected cumulative rewards (also called *return*).
The environment is typically stated in the form of a *Markov decision process (MDP)* that allows many theoretical guarantees. Many reinforcement learning algorithms use dynamic programming techniques or are inspired by them. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP, and they target large MDPs where exact methods become infeasible. In RL, the agent has to find a balance between exploration (better knowing the environment) and exploitation (using current knowledge) [34].
Modern RL algorithms seek either to approximate the value ($\mathcal{Q}$ function) or directly to find the best policy π_θ that maximizes the discounted return by adjusting the parameters θ (typically neural nets' parameters).
To apply RL algorithms for our case, we need to define our problem as a RL environment with observations, actions and rewards.

Environment : When using RL, we tackle MFVA with omission assignment, since it has a smaller action space $\mathcal{A}$: it is the discrete space $[d]$, an action $i \in \mathcal{A}$ means the i-th package is omitted. Let us denote by $R(i)$ the subrouting of R, where destination i is removed from the route where it appears in R. When action is taken, the current routing R is replaced by $R(i)$.
We consider episodes, which are agent-environment interactions from initial to final states. In our context, an episode is on a fixed instance I of MFVA and consists of removing destinations until the quota is respected. The initial state is the instance I and the routes generated by the routing layer with all destinations included, and the excess emission $\mathrm{em}(R) - Q$. Accordingly, the final state is the state in which the excess emission is negative or null. Thus, we consider a large MDP, constituted of multiple independent MDPs, each of them dealing with a particular instance I. The objective, however, is to find some regularity and common pattern among these MDPs to be able to learn something which could work over all instances.
The state of our MDP consists in the instance I and the current routes. Therefore, the state space of this MDP encompasses all possible instances and routes, which represents a complex and heavy observation for learning algorithms to process. As a result, we need to select relevant observations for the learner agent to interact with this environment. This leads us into the domain of Partially Observable MDPs (POMDPs) [34].
We construct an informative observation vector o by incorporating the λ-penalized costs based on emissions, as described in Section 3. We also add to the state the route of each

vehicle and the excess emissions $(\mathrm{em}(R) - Q)^+$. So, we have $\dim(o) = (d+1)^2 + |\mathcal{V}|(\max_{v\in\mathcal{V}} Cap_v + 2) + 1$, thus, we define the observation space as $\mathcal{O} = \mathbb{R}^{\dim(o)}$.
We also tested alternative observation spaces. To expedite the learning process, we may provide the agent with lighter observations, reducing the dimensionality. For instance, instead of giving the entire cost matrix, we could provide only the routes along with the cost of transitioning from the previous destination to the next, along with information about excess emissions. However, neither of these state spaces led to improved performance for our learning agents.
We tested other observation spaces to reduce dimensionality by providing the agent with lighter observations. For example, instead of providing the entire cost matrix, we provide only the routes along with the cost of going from the preceding destination to the next, accompanied by information about excess emissions. But no alternate state space has improved the performances of our learning agents.
Let i be the agent's action and R the current routing on instance I. Recall that $R(i)$ is the subrouting induced by the elimination of the destination i from the routing R and $q(i)$ be the quantity demanded by the destination i, the rewards are computed as $\frac{P\sum_{j\in[d]} q(j) - \mathcal{L}_P(R(i))}{P\sum_{j\in[d]} q(j)}$ if $(\mathrm{em}(R(i)) - Q)^+ > 0$ and 0 otherwise. $\mathcal{L}_P$ is the loss function defined before. Hence, the reward represents the relative gain over the cost of omitting all packages in range $[0, 1]$.

RL Model : The state-of-the-art algorithms, such as *Proximal Policy Optimization* (PPO) [31], which we use in this work, operate within an *actor-critic* framework. This actor-critic method involves approximating both the actor function π and the value function V, which serves as the critic in the algorithm. Typically, parameterized functions π_θ and V_σ are employed, with θ and σ denoting the parameters of each function. Modern reinforcement learning (RL) techniques leverage neural networks and deep learning to optimize these parameters, which is why they are often referred to as *deep reinforcement learning* methods.
In our implementation, we employ a five-layer Multilayer Perceptron (MLP) network to generate a feature vector. The parameters of these layers are shared between the actor and value functions. Each function then utilizes this feature vector to produce either the value or the probability distribution over actions through a straightforward linear regression process. The parameters are updated based on the rewards obtained in each episode.
In our model, since selecting a previously executed action has no impact, we incorporate a mask to exclusively consider remaining actions. This variant is known as Maskable PPO [15], where the probability of choosing invalid actions is set to zero, and the action is selected according to the renormalized probability distribution of the remaining actions. This avoids unnecessary actions, thereby accelerating the learning process.

Multi-agent Learning

We solve the MFVA in a multi-agent model, in which each package is an agent. To make the routing layer fast enough, routes are calculated by the Nearest Neighbor algorithm described in Section 3. Each package k is an agent that chooses its vehicle $a_k \in \mathcal{V}\cup\{0\}$.

Then, they either observe their collective reward $r(a)$, or an individual one $r_k(a)$ with $a = (a_1, \ldots, a_K)$. Let $D^\lambda[v] = D_v^\lambda$, with D_v^λ the penalized cost matrix constructed in Section 3. Let $a(i)$ be the attributed vehicle of the destination/package i, prec(i) be the precedent destination before i, and next(i) the next destination after i in the route $R_{a(i)}$. Let Δ be the marginal gain function when we omit a destination, as defined in Section 4. We define the marginal cost vector δ as

$$\delta_i = \Delta(a(i), \text{prec}(i), \text{next}(i))$$

Let LCF(v) be the *local cost function* that represents the total cost of the vehicle v's tour with the cost matrix D_v^λ and the omission penalties of that vehicle. The omission penalty is added when the capacities required by the packages assigned to the vehicle v outnumber the capacity of the latter. Formally,

$$\text{LCF}(v) = \text{cost}(R_v) + \text{Oq}(R_v)P + \lambda\, \text{em}_v(R_v)$$

So we have, with the assignment/action vector a,

$$\mathcal{L}_P^{(i)}(a) = -\left(\frac{\delta_i}{\max_j \delta_j} + \frac{\text{LCF}(a(i)) - \min_j \text{LCF}(j)}{\max_j \text{LCF}(j) - \min_j \text{LCF}(j)}\right)$$

In the multi-agent case, we only consider the vehicle assignment. If the number of packages assigned to a vehicle exceeds its capacity, the packages with higher indices are omitted, and a penalty is added to the LCF cost. This penalty impacts all packages in the affected vehicle.

We consider two classical methods for the multi-agent learning, LRI [30] and EXP3 [2]. These algorithms operate *without any contextual information or observation*, focusing solely on learning the optimal policy through observations of the rewards obtained from their actions. Each agent i receives the reward defined as $-\mathcal{L}_P^{(i)}(a)$. This reward accounts for both individual and collective performance. This reward is normalized and is in $[-2, 0]$ which helps with the convergence of our decentralized learning algorithms to a relevant solution.

5 Experimental Results

For the experimental results[2], we consider different scenarios with varying numbers of unique destinations. First, we compare the performance of the methods when all packages i have the same quantity $q(i) = 1$ for 100 destinations. Next, we evaluated the methods with varying quantities for each package, for 20 destinations only. The instances are generated randomly, following a distribution derived from real delivery data in France (see details in appendix C).

First, we evaluate all methods except the RL approach to identify the best baseline methods for comparison with the RL method. We run the algorithms over 100 different randomly chosen instances and compare their relative performance against the optimal

[2] The code is available online at `https://github.com/Farid-Najar/TransportersDilemma`.

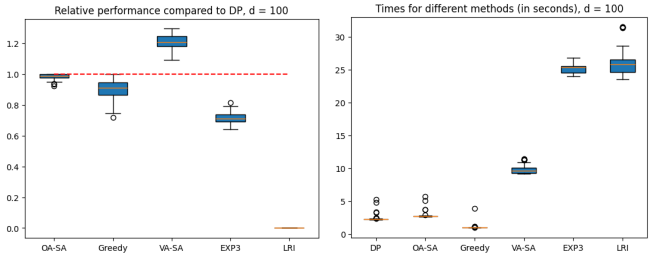


Fig. 1: The performance of different methods over 100 instances with the same parameters. On the vertical axis, the rewards relative to a reference algorithm (higher is better) or execution times (lower is better).

algorithm for the SHORTCUT problem, i.e., dynamic programming. For the $d = 20$ scenario, we exclude LRI from the experiments due to its poor performance in previous tests and its consistent underperformance compared to EXP3. In these experiments, the routes are fixed for algorithms working with omission assignments, effectively solving the simpler SHORTCUT problem.

We begin with methods that make omission assignments on fixed routes and compare them with vehicle assignment methods in Figures 1. The comparison is made using the reward defined for RL methods. The results indicate that dynamic programming (DP) is the most efficient among the omission assignment methods, while VA−SA outperforms all other methods. This highlights the importance of adjusting routes in QVRP to construct optimal solutions. This phenomenon is further illustrated in Figures 10 and 11 of the appendix: the routing in classical VRPs, which partitions the destinations, differs significantly from the best routing for QVRP, where less pollutant vehicles are required to take longer routes in order to respect the emission quota.

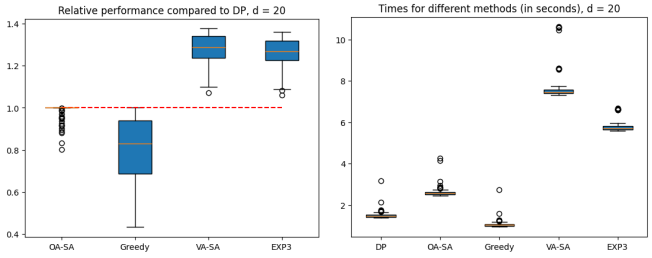


Fig. 2: The performance of different methods for $d = 20$ with destinations that have different quantities over 100 instances with same parameters.

All algorithms demonstrated reasonable execution times. The greedy removal method also yielded relatively good results while being faster than the other methods. On the other hand, LRI failed to find a valid solution, whereas EXP3 was able to converge

to a valid solution, achieving results comparable to `VA-SA` in the $d = 20$ case. This is notable considering that EXP3 agents have limited information about the instances, relying solely on the rewards received from their interactions with the environment, and their routes are suboptimal in terms of distance.

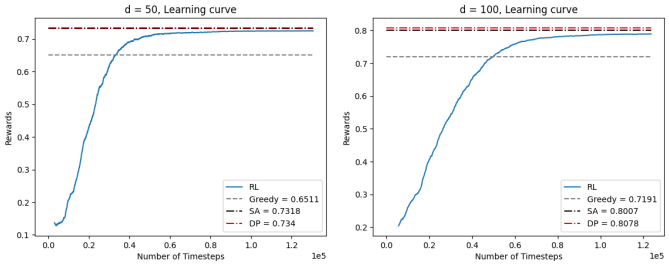


Fig. 3: The performance of the RL agent compared to other methods. The experiments are done on a single instance, with the same routes and destinations at every episode. The learning curves is the mean over 10 different instances picked randomly.

For the RL agents, we first trained them on one instance to see if they were able to learn in this case. As we can see in Figure 3, the RL agent converged to a solution nearly as good as the one given by DP surpassing `Greedy`.

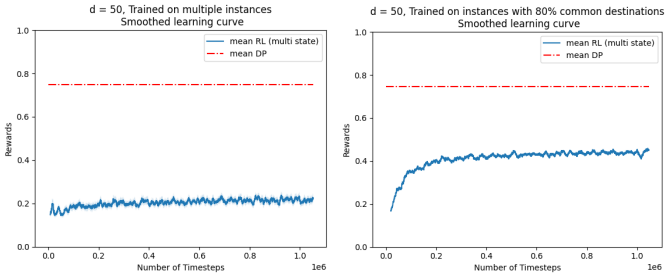


Fig. 4: The performance of the RL agent trained on 200 different instances compared to the average performance of DP in the scenario with $d = 50$. On the right-hand side, we keep 80% of destinations unchanged in all instances.

We also trained the RL agent on a variety of instances, hoping it would generalize to all instances with a given number of destinations. However, it failed to learn anything, as shown on the left side of Figure 4. When trained on a dataset where 80% of the destinations remained unchanged, the RL agent was able to learn, although it still could not outperform the solution found by DP. Specifically, it learned to route the 80% of destinations that remained constant, but it was unable to adapt to the changing destinations.

6 Discussion

We addressed the demand selection problem in QVRP, a VRP with a strict emission constraint and the option to omit deliveries to comply with this constraint. Specifically, we studied MFVA, an assignment problem interacting with a QVRP.
We developed and compared several methods based on classical OR and machine learning (ML). Our experiments showed that OR methods, particularly simulated annealing, outperformed ML approaches in terms of efficiency. The reinforcement learning (RL) agent failed to surpass classical algorithms on large instances, and only achieved similar results on small instances at the cost of increased computation time. This finding aligns with prior studies [5,22,20,8,13,23]. Additionally, RL struggled to generalize across all problem instances.
Decentralized learning algorithms, such as EXP3, demonstrated promising results given their limited problem knowledge but remained slower and less effective than simulated annealing.
Our results suggest that pure learning and end-to-end approaches may not be well-suited for combinatorial problems, at least in static settings. Future work will focus on extending these methods to dynamic variations of the problem, where learning-based approaches are expected to perform better.

Acknowledgments. This work was supported by the French LabCom HYPHES (ANRS-21-KCV1-0002).

References

1. Applegate, D., Bixby, R., Chvatal, V., Cook, W.: Concorde tsp solver (2006)
2. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM journal on computing **32**(1), 48–77 (2002)
3. Barth, D., Cohen, J., Echabbi, L., Hamlaoui, C.: Transit prices negotiation: Combined repeated game and distributed algorithmic approach. In: International Conference on Network Control and Optimization. pp. 266–275. Springer (2007)
4. Bektaş, T., Laporte, G.: The pollution-routing problem. Transportation Research Part B: Methodological **45**(8), 1232–1250 (2011)
5. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural Combinatorial Optimization with Reinforcement Learning (Jan 2017)
6. Chaharsooghi, S.K., Heydari, J., Zegordi, S.H.: A reinforcement learning model for supply chain ordering management: An application to the beer game. Decision Support Systems **45**(4), 949–959 (2008), information Technology and Systems in the Internet-Era
7. Cunha, B., Madureira, A., Fonseca, B., Matos, J.: Intelligent scheduling with reinforcement learning. Applied Sciences **11**(8), 3710 (2021)
8. Eberhard, O., Cuvelier, T., Valko, M., Backer, B.D.: Middle-Mile Logistics Through the Lens of Goal-Conditioned Reinforcement Learning. NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning (2023)
9. Eslamipoor, R.: Direct and indirect emissions: a bi-objective model for hybrid vehicle routing problem. Journal of Business Economics **94**(3), 413–436 (2024)
10. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 174. freeman San Francisco (1979)

11. Garside, A.K., Ahmad, R., Muhtazaruddin, M.N.B.: A recent review of solution approaches for green vehicle routing problem and its variants. Operations Research Perspectives p. 100303 (2024)
12. Glover, F.W., Kochenberger, G.A.: Handbook of metaheuristics, vol. 57. Springer Science & Business Media (2006)
13. Habib, Y., Filchenkov, A.: Multi-agent reinforcement learning for multi vehicles one-commodity vehicle routing problem. Procedia Computer Science **212**, 418–428 (2022)
14. Helsgaun, K.: An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. Roskilde: Roskilde University **12**, 966–980 (2017)
15. Huang, S., Ontañón, S.: A closer look at invalid action masking in policy gradient algorithms. In: The International FLAIRS Conference Proceedings. vol. 35 (2022)
16. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Operations research **37**(6), 865–892 (1989)
17. Kool, W., van Hoof, H., Gromicho, J., Welling, M.: Deep policy dynamic programming for vehicle routing problems. In: International conference on integration of constraint programming, artificial intelligence, and operations research. pp. 190–213. Springer (2022)
18. Kool, W., van Hoof, H., Welling, M.: Attention, learn to solve routing problems! In: ICLR (2019)
19. Labidi, H., Azzouna, N.B., Hassine, K., Gouider, M.S.: An improved genetic algorithm for solving the multi-objective vehicle routing problem with environmental considerations. Procedia Computer Science **225**, 3866–3875 (2023)
20. Li, S., Yan, Z., Wu, C.: Learning to delegate for large-scale vehicle routing. Advances in Neural Information Processing Systems **34**, 26198–26211 (2021)
21. Lou, P., Zhou, Z., Zeng, Y., Fan, C.: Vehicle routing problem with time windows and carbon emissions: a case study in logistics distribution. Environmental Science and Pollution Research pp. 1–21 (2024)
22. Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., Tang, J.: Learning to iteratively solve routing problems with dual-aspect collaborative transformer. Advances in Neural Information Processing Systems **34**, 11096–11107 (2021)
23. Mak, S., Xu, L., Pearce, T., Ostroumov, M., Brintrup, A.: Fair collaborative vehicle routing: A deep multi-agent reinforcement learning approach. Transportation Research Part C: Emerging Technologies **157**, 104376 (2023)
24. Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E.: Reinforcement learning for combinatorial optimization: A survey. Computers & Operations Research **134**, 105400 (2021)
25. Molina, J.C., Eguia, I., Racero, J., Guerrero, F.: Multi-objective vehicle routing problem with cost and emission functions. Procedia - Social and Behavioral Sciences **160**, 254–263 (2014), xI Congreso de Ingenieria del Transporte (CIT 2014)
26. Nazari, M., Oroojlooy, A., Snyder, L., Takác, M.: Reinforcement learning for solving the vehicle routing problem. Advances in neural information processing systems **31** (2018)
27. Perron, L., Furnon, V.: Or-tools (2024), https://developers.google.com/optimization/
28. Pilati, F., Tronconi, R.: Multi-objective optimisation for sustainable few-to-many pickup and delivery vehicle routing problem. International Journal of Production Research **62**(9), 3146–3175 (2024)
29. Qin, G., Tao, F., Li, L.: A vehicle routing optimization problem for cold chain logistics considering customer satisfaction and carbon emissions. International journal of environmental research and public health **16**(4), 576 (2019)
30. Sastry, P.S., Phansalkar, V.V., Thathachar, M.: Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information. IEEE Transactions on systems, man, and cybernetics **24**(5), 769–777 (1994)

31. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017), arXiv:1707.06347 [cs]
32. Shi, R., Niu, L.: A brief survey on learning based methods for vehicle routing problems. Procedia Computer Science **221**, 773–780 (01 2023)
33. Stenger, A., Schneider, M., Goeke, D.: The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. EURO Journal on Transportation and Logistics **2**(1-2), 57–87 (2013)
34. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
35. Talbi, E.G.: Machine learning into metaheuristics: A survey and taxonomy. ACM Computing Surveys (CSUR) **54**(6), 1–32 (2021)
36. Vidal, T.: Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. Computers & Operations Research **140**, 105643 (2022)
37. Zhang, J., Zhao, Y., Xue, W., Li, J.: Vehicle routing problem with fuel consumption and carbon emission. International Journal of Production Economics **170**, 234–242 (2015)

A Limits to the use of SHORTCUT as a proxy for QVRP

Hardness of Shortcut

We prove that when all vehicles have different cost and emission functions, SHORTCUT is NP-hard to solve. Hence, the only theoretical question left is whether there is an FPT algorithm with parameter the number of different types of vehicles.

Theorem 2. SHORTCUT *is* NP*-hard.*

Proof. We consider the KNAPSACK problem, where we are given $(C, V, k, (c_i, v_i)_{i\in[n]})$, and we must decide whether there is a set of indices I, such that $\sum_{i\in I} c_i \leq C$ and $\sum_{i\in I} v_i \geq V$. Since KNAPSACK is NP-complete [10], we give a Turing reduction from KNAPSACK to SHORTCUT to prove its NP-hardness. We build a routing $R = (R_1, \ldots, R_n)$ such that $R_i = (d_0, d_i, d_0)$ and the destination d_i is chosen to be at distance $v_i/2$ of d_0, hence the length of R_i is equal to v_i. We set all emission factors of the vehicles to 1, the quantity of each destination to 1 and the quota to $Q = \sum_{i\in[n]} v_i - V$. Therefore, if we consider a set $I \subseteq [n]$ of elements, and we remove the single destination in the route R_i for $i \in I$, the quota is satisfied if and only if $\sum_{i\in I} v_i \geq V$.
We let B be the maximum of the c_i. We set the cost factor of the vehicle following R_i to $\frac{B-c_i}{2v_i}$, such that the cost of R_i is equal to $B - c_i$. Hence, solving SHORTCUT for the instance we have built and k the number of destinations to remove, gives us, if it exists, I of size k which minimizes the cost of $R(I)$ and satisfies $\sum_{i\in I} v_i \geq V$. Minimizing the cost of $R(I)$ means maximizing the length removed, that is $\sum_{i\in I} B - c_i$. It is thus equivalent to minimizing $\sum_{i\in I} c_i$ since I is of fixed size k and B a constant. Therefore, if it is possible for $\sum_{i\in I} c_i \leq C$, then the solution given by solving SHORTCUT satisfies this constraint. Therefore, solving the SHORTCUT instance we have built for all k, we can decide whether the original instance of KNAPSACK has a solution.

Quality of SHORTCUT solution as a heuristic for QVRP

Here, we prove that using a solution to the SHORTCUT problem as a heuristic to solve QRVP may be very far from optimal.
We consider an instance with n destinations $d_1, \ldots, d_n$ each of quantity 1 and their distances from the hub d_0 are $D[0, i] = 2^i$. We have n vehicles of capacity 1, the emission factor of vehicle i is 2^{-i} for all $i \in [n]$ and the cost factors are equal to 1.
When we produce a good first routing without removing destinations in the routing layer, we try to minimize the emission. Here, the routing R where $R_i = (d_0, d_i, d_0)$ has an emission of 1 for each route. Hence, we have $e(R) = n$, and this routing has minimal emission. We let the quota be $Q = n/2$. If we solve the problem SHORTCUT on this instance, since we must minimize the number of elements to remove to respect the quota, we have to remove the destinations of exactly $n/2$ vehicles. To minimize the cost, we must remove the destinations of the $n/2$ last vehicles which have the largest cost.
If we solve optimally the problem QVRP on the same instance, we get a different solution. It is always better to remove the destinations with the largest distance to the hub and to avoid the first vehicles with the largest emission functions. Hence, if we remove

the last destination, and we do not use the first vehicle, we have a routing of emission $(n-1)/2 < Q$. Therefore, the optimal solution of QVRP only requires removing a single destination, while the solution to SHORTCUT requires removing half the destinations. Hence, using a solution to SHORTCUT as a heuristic to solve QVRP can be arbitrarily bad, even though in experiments it gives reasonable results, see Section Experiments.
By modifying the cost factor, it is also possible to prove that the gap in cost can be arbitrarily large. We could also prove smaller gaps in the number of packages to remove when the cost and emission factors are all within some fixed range.

B Details On RL

RL methods are implemented using `Stable Baselines 3`. The value function and the policy function share the same NN parameters for the feature function, upon which we add a last independent linear layer for each function. The feature function's model is a simple feed forward linear NN in 5 layers with sizes [4096, 2048, 2048, 1024, 512] respectively for each layer. We use the ReLU function as the non-linear activation function. The optimization process is done using the Adam optimizer. The rest of the hyperparameters are by default parameters of the Maskable PPO in `Stable Baselines 3`.
For the results of the RL on one instance (as shown in the submitted article), we take a particular instance and train the RL agent over the same instance at every episode. On the other hand, in the experiments of the RL for multiple instances, we change the instance at every episode.

C Details On Methods and Experiments

The code to reproduce the experiments is available online[3]. Experiments are done on a MacBook Pro M1 with 16Go of RAM.

Real data We generate 200 instances for each scenario. The destinations are drawn from the anonymized data provided by a company. The frequency of the deliveries are used as a probability vector to draw destinations. The distance matrix is computed using the longitude and the latitude of the destinations. The emission quota Q is fixed arbitrary equal to 125 for $d = 100$, 100 for $d = 50$ and 75 for $d = 20$.

Synthetic data We generate 200 instances for each scenario. The destinations are drawn uniformly at random on a plane of dimension 12×12. The distances between two nodes are then computed using the Euclidean distance in the plane. The emission quota Q is fixed arbitrary equal to 10 for $d = 20$, and 20 otherwise.

[3] `https://github.com/Farid-Najar/TransportersDilemma`

Shared between synthetic and real data Our instances have four vehicles[4]: one electric vehicle of emission factor 0, one hybrid of emission factor 0.15 and 2 diesels of emission factor 0.3. All vehicles have the same capacity, which is chosen with respect to the number of vehicles and the total capacity necessary for delivering all packages.
Among generated instances, we discard those that respect the quota without the need of omitting packages to better discriminate the efficiency of our algorithms. Note that quota determines the degree of difficulty of the assignment problem. Indeed, the more strict is this constraint relative to the total emissions of the delivery operation, more packages need to be omitted to respect the constraint.
For $d = 20$, we chose quantities as $q = \mathbf{1}_d + (C * M - d) * \lfloor X \rfloor$ with C the capacity of a vehicle, M the number of vehicles, and $X \sim \text{Dir}(\alpha = \mathbf{1}_d)$. We redraw X if the total quantity demanded exceeds the total capacity.

Hyperparameters

The emission penalty λ is set to 10000[5] to be high enough to penalize effectively the emissions. It biases the solution found by the routing layer so that it minimizes the emissions to hopefully respect the quota.

Table 1: Hyperparameters.

Hyperparameters	**Values**
Simulated Annealing	
τ_{init}	5000
τ_{limit}	1
cooling rate	0.995
EXP3	
η	$1/d\sqrt{t}$
γ	0.1
LRI	
b	0.003
or-tools	
Initial solution strategy	Nearest neighbor
Local search algorithm	GLS
Execution time limit	10s for $d = 20$, 60s for $d = 50$, 120s for $d = 100$

[4] Vehicles' emissions are chosen to have a diverse set of characteristics.
[5] Chosen arbitrarily, but we could tune it to be relevant for the real emission targets considered by cities in the future.

Additional Figures

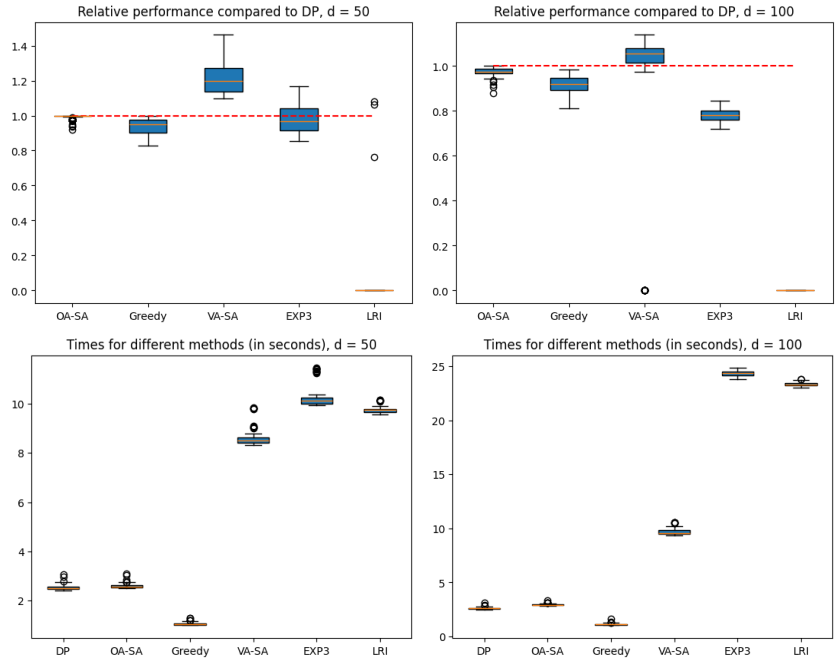


Fig. 5: The performance of different methods using synthetic data. On the vertical axis, the rewards relative to a reference algorithm (higher is better) or execution times (lower is better).

Comments on the routes As we can see in the Figures 10 and 11, the routes generating the best rewards are those that make the EV and hybrid trucks travel long distances and, on the other hand, the diesel vehicles does much less in distance. This seems natural for our problem, but it is in contrast with the usual VRP routings that tries to distribute the destinations, by partitioning the destinations into zones with roughly the same number of destinations. Then, each zone is served by one vehicle.

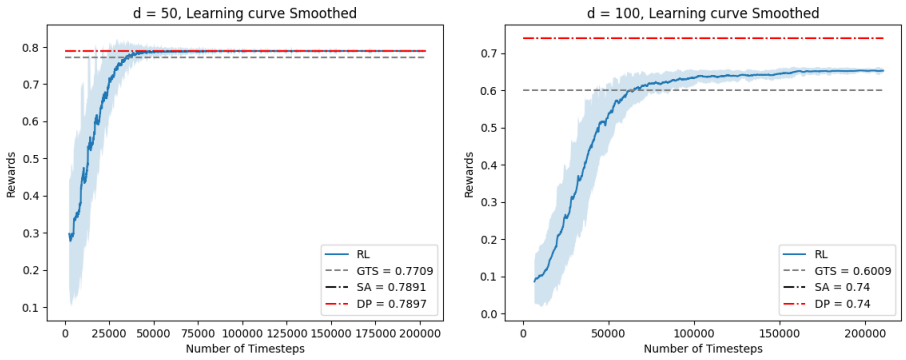


Fig. 6: The performance of the RL agent, using synthetic data, compared to other methods. The experiments are done on a single instance with the same routes and destinations at every episode. (Areas in light color indicate the standard deviation of values.)

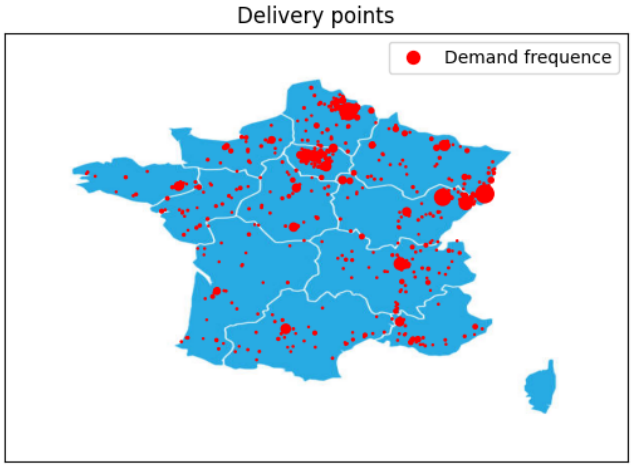


Fig. 7: Demands and their frequencies of the real data of the deliveries in France on the map.

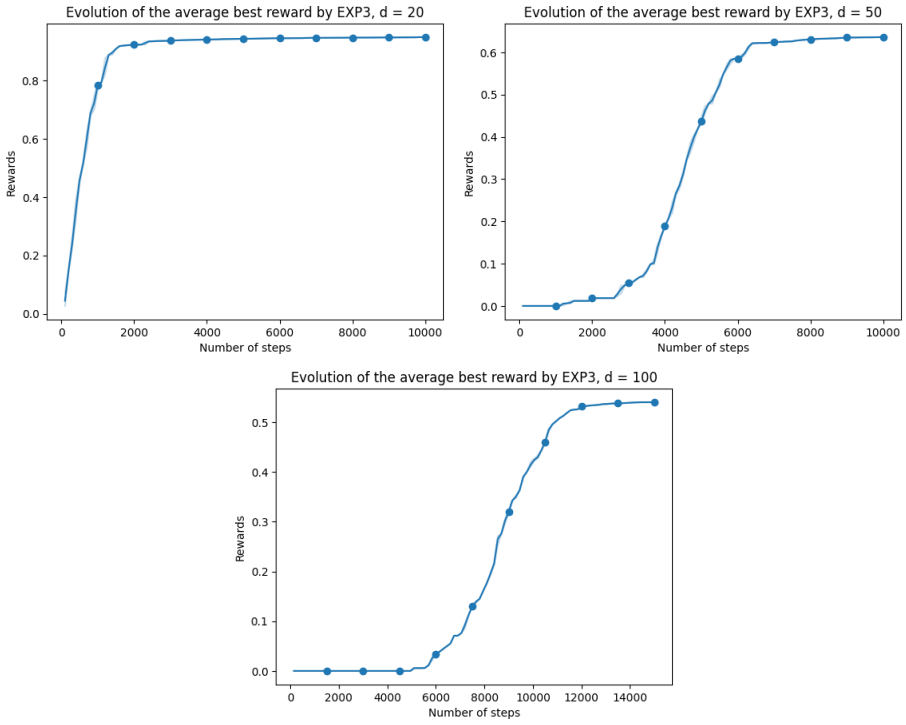


Fig. 8: The average best reward found by EXP3 per step. The average is done over 100 experiments. We can remark the convergence of the EXP3 algorithm towards some solution.

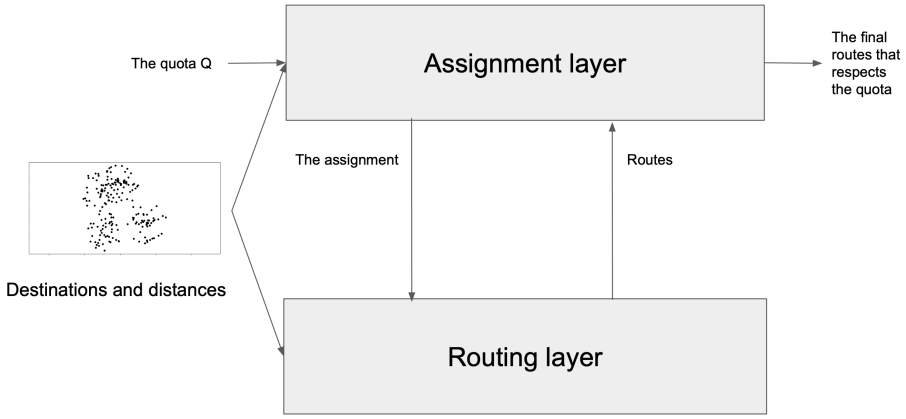
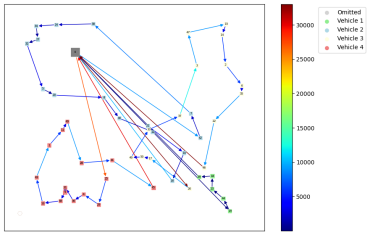
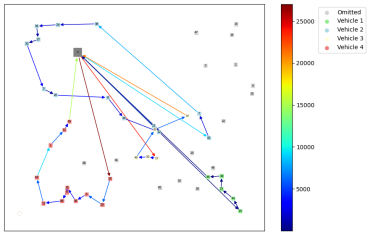


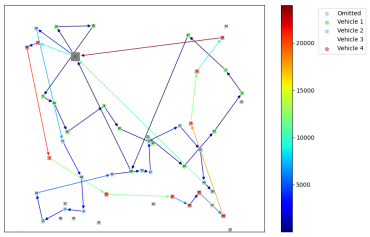
Fig. 9: This Figure shows how different layers work and interact with each other.



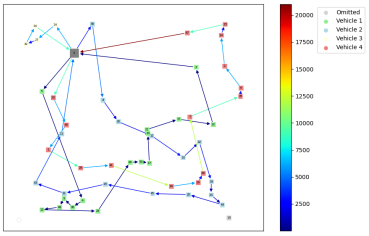
(a) The initial routes given by `or-tools`



(b) Routes of the DP solution

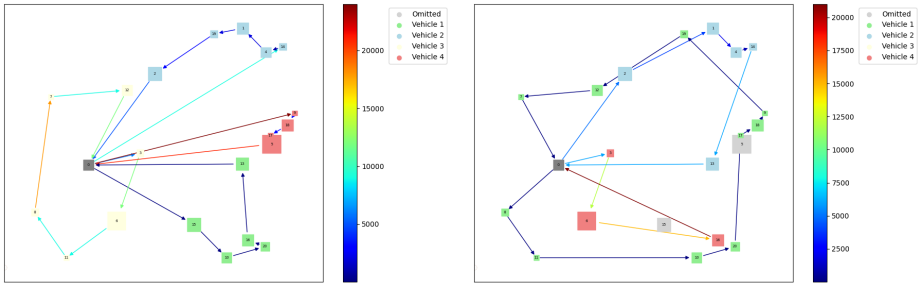


(c) Routes of the multi-agent EXP3 solution

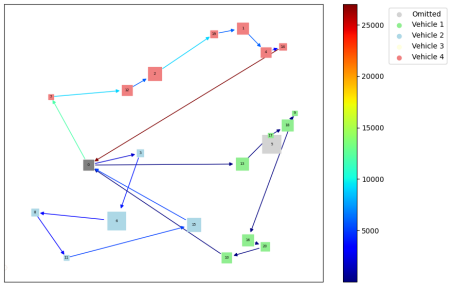


(d) Routes of the `VA-SA` solution

Fig. 10: Routes given by different methods for $d = 50$. Node colors indicate the vehicle, and the color heatmap of each arc indicates the cost of that arc. Vehicle 1 is EV, 2 hybrid, and 3 & 4 are diesel with the same characteristics.



(a) The initial routes given by `or-tools` (b) Routes of the multi-agent EXP3 solution



(c) Routes of the `VA-SA` solution

Fig. 11: Routes given by different methods for $d = 20$. Node colors indicate the vehicle, their size indicate the quantity demanded by that destination, and the color heatmap of each arc indicates the cost of that arc. Vehicle 1 is EV, 2 hybrid, and 3 & 4 are diesel with the same characteristics.