Deep reinforcement learning for stochastic last-mile delivery with crowd shipping

Marco Silva, João Pedro Pedroso, Ana Viana

Deep reinforcement learning for stochastic last-mile delivery with crowd shipping

Marco Silva[a,*], João Pedro Pedroso[b], Ana Viana[c]

[a] *INESC TEC, Porto, Portugal*
[b] *INESC TEC and Universidade do Porto, Porto, Portugal*
[c] *INESC TEC and Polithecnic of Porto, Porto, Portugal*

Abstract

We study a setting in which a company not only has a fleet of capacitated vehicles and drivers available to make deliveries but may also use the services of occasional drivers (ODs) who are willing to make deliveries using their vehicles in return for a small compensation. Under such a business model, a.k.a crowd shipping, the company seeks to make all the deliveries at the minimum total cost, i.e., the cost associated with their vehicles and drivers plus the compensation paid to the ODs.

We consider a stochastic and dynamic last-mile delivery environment in which customer delivery orders, as well as ODs willing to make deliveries, arrive randomly throughout the day and present themselves for deliveries made within fixed time windows.

We present a novel deep reinforcement learning (DRL) approach to the problem that can deal with large real-life problem instances, where we formulate the action selection problem as a mixed-integer optimization program.

The DRL approach is compared against other approaches to optimization under uncertainty, namely, sample-average approximation (SAA) and distributionally robust optimization (DRO). The results show the effectiveness of the DRL approach by examining out-of-sample performance and that it is suitable to process large samples of uncertain data.

Keywords: Last-mile delivery, Crowd shipping, Deep reinforcement learning, Data-driven Optimization, Integer Optimization

1. Introduction

Last-mile delivery is defined as the movement of goods from a transportation depot to the final delivery destination, which is typically a personal residence.

*Corresponding author
Email addresses: marco.c.silva@inesctec.pt (Marco Silva), jpp@fc.up.pt (João Pedro Pedroso), ana.viana@inesctec.pt (Ana Viana)

The worldwide growth of e-commerce caused the increase of importance and competitive value of the last-mile delivery and prompted many companies to seek creative and innovative solutions. Among those, there is the crowd sourcing of some orders to third-party couriers, or occasional drivers (ODs).

Crowd shipped delivery has been adopted as a shortcut to last-mile growth. It has been implemented under different business models depending on how the occasional drivers are engaged and managed. A survey in ARC Advisory Group (2021) indicates that while only 9% of retailers are using crowd sourced providers now, one in four retailers plans to start using them in the next 12 months. It has been implemented as an enabler to same-day delivery for the last mile as can be seen in recent implementations of large companies as in Walmart (2021), Doordash (2021) and JD-Dada (2021).

The advantages of crowd shipping are numerous and are not only related to economic issues, due to the compensation for the ODs being potentially less than the cost associated with delivering using their own capacitated vehicles. If relying on the idea of individuals sharing their potentially under-utilized property, sharing vehicles can lead to a reduction in polluting emissions, energy consumption, noise and traffic congestion.

The application of crowd shipping alluded to above gives rise to new variants of the routing problem. It has been addressed as an extension of the classical vehicle routing problem (VRP) or traveling salesman problem, modeled under different deterministic, stochastic and dynamic optimization approaches.

In this paper, we consider a setting in which a company not only has a fleet of capacitated vehicles and drivers available to make deliveries, but may also use the services of ODs who are willing to make deliveries using their vehicles in return for a small compensation. Under such business model, a.k.a. crowd shipping, the company seeks to make all the deliveries at a minimum total cost, i.e., the cost associated with its vehicles and drivers plus the compensation paid to the ODs. Our setup is suitable for a same-day delivery scheme where time windows are fixed, predefined periods during the day, and customers with online orders and available occasional drivers can enlist themselves to these time windows.

We adopt a data-driven dynamic and stochastic approach where the existence of online customers orders to be delivered, as well as the availability of occasional drivers to deliver them are random and define scenarios on which decisions have to be made.

This problem is complex because decisions, regarding the dispatch of vehicles or occasional drivers, have to be made fast and the space to search for decisions is potentially too large. We then propose a deep reinforcement learning (DRL) method where we model the problem as a sequence of states connected by actions, driven by decisions, and transitions. The DRL method uses a neural network (NN) as approximation architecture for the problem value function. Our approach is data-driven and we assume there are scenarios available as historical data that is used to train the NN.

Another key feature of our DRL approach is how we search the action (decision) space. Most reinforcement learning (RL) studies on stochastic VRPs face

the challenge imposed by the combinatorial nature of state and action spaces by restricting the action space and aggregating the state space based on expert knowledge. Here, we formulate the action selection problem for each state using a recourse in a two-stage decision model where the first-stage decision is formulated as a mixed-integer optimization program. At the first stage, only the order in which all customers will be delivered is established. The second-stage decision is made every time a scenario is revealed, and before any dispatch of fleet vehicles or ODs. The second-stage decision comprises routes defined by the recourse, where the routes follow the first-stage decision ordering but skip customers that have no online orders or customers outsourced for available ODs. Each time the vehicle capacity or the time window limit is reached, a return path to the depot is created and another route restarts from the depot if needed.

The main contributions of the approach above and results of this work are:

- We propose a novel data-driven stochastic and dynamic approach for crowd shipping last-mile delivery, advancing the state-of-the-art in this topic. Different from most stochastic crowd shipping routing studies in the literature, uncertainty is only related to customer events (i.e. customers with delivery orders or with ODs available to outsource their orders), and this reduces complexity of the solution methods involved.
- We experiment with a method that integrates machine learning and optimization techniques. By doing this we can extend a recourse model coming from stochastic optimization to a RL application and introduce optimization techniques to search the RL action space.
- We provide computational evidence on the capability of the proposed model, w.r.t. a more realistic assumption of correlated scenarios, to obtain solutions that can improve those provided using more simplified assumptions. In particular, we compare the DRL algorithm results with the ones provided by a distributionally robust optimization approach to the same problem, where we search the optimal solution considering a worst case joint probability distribution for the scenarios, and with the ones provided by a sample-average approximation method.

In what follows, in Section 2, we review relevant approaches to solve variants of the problem, and that can be used to contextualize our approach. In Section 3, we formally present our problem and the model we have defined. Section 4 introduces the DRL method developed. Next, in Section 5, we present and discuss the computational results. Finally, in Section 6, we present the conclusion of the work done.

2. Literature review

In the following sections we survey relevant literature for the proposed approach. It includes not only the publications related to models for the crowd shipping of the last-mile delivery, but also approaches for the problems where the customers are uncertain and applications of RL methods to VRPs.

2.1. Crowd Shipping Routing

A seminal work on last-mile delivery with crowd shipping is presented in Archetti et al. (2016). The authors study a deterministic approach where the customers' locations and the ODs' parameters are input data. The model proposed is a combination of an assignment problem, where ODs are assigned to customers, with a capacitated VRP where routes are defined for vehicles passing through customers not served by ODs. The pricing mechanism, meaning how compensation fees are defined, constitutes a critical part of the model and is discussed in more detail by the authors. The authors develop a multi-start heuristic to handle instances with more than 25 customers.

The work in Archetti et al. (2016) was later extended by other authors considering time windows with multiple and split deliveries (Macrina et al. (2017)), transshipment points (Macrina et al. (2020)), and the situation where occasional drivers on bikes or on foot are coordinated with a delivery truck from which they relay (Kafle et al. (2017) and Huang and Ardiansyah (2019)) . All the above mentioned papers consider deterministic versions of the problem.

Differently, in Arslan et al. (2019), the authors develop a dynamic solution alternative, where the solution is adjusted every time new information is available. They consider a service platform that automatically creates matches between parcel delivery tasks and ODs. The matching of tasks, drivers, and dedicated vehicles in real-time gives rise to a new variant of the dynamic pickup and delivery problem.

The authors in Dayarian and Savelsbergh (2020) introduce a stochastic and dynamic routing problem in which the demand arrives over time, as also does part of the delivery capacity, in the form of in-store customers willing to make deliveries. They develop two rolling horizon dispatching approaches to the problem: one that considers only the state of the system when making decisions, and one that also incorporates probabilistic information about future online orders and in-store customer arrivals. Random events are considered independent in their approach.

In Dahle et al. (2017), the authors consider stochastic ODs and define routes for the company vehicles and the ODs based on their destination. They consider time windows for when the ODs appear and use a two-stage model to define only partial routes in the first stage. The stochastic solution is based on a scenario approach, and they assume a uniform distribution of scenarios. Results are reported on instances with up to 20 customers and three ODs.

In Archetti et al. (2021) the authors investigate an online vehicle routing problem with ODs in which customer requests are either known in advance with respect to the planning of the distribution, or they arrive online during the distribution process. Each request and OD is associated with a time window and penalties are incurred when ODs violate time windows as well as when a request is not served.

In Gdowska et al. (2018), the authors consider that customers can be offered or not to potential ODs and that there is a known probability of them being accepted. They develop a heuristic to identify which customers will be offered to

Paper	Uncertainty	Customers	ODs	VRP extension
Archetti et al. (2016)	Deterministic	Constant fixed set	OD parameters as input	-
Macrina et al. (2017)	Deterministic	-	-	Time windows, multiple and split deliveries
Macrina et al. (2020)	Deterministic	-	-	Transshipment points
Kafle et al. (2017) and Huang and Ardiansyah (2019)	Deterministic	-	-	Delivery truck relay
Arslan et al. (2019)	Deterministic and Dynamic	on line	on line	Pickup and delivery
Dayarian and Savelsbergh (2020)	Stochastic and Dynamic Events are independent	on line	on line	-
Dahle et al. (2017)	Stochastic Scenario based	Constant fixed set	OD parameters as input	Time windows
Archetti et al. (2021)	Deterministic and Dynamic	on line	on line	Time windows
Gdowska et al. (2018)	Stochastic and Dynamic Events are independent	Constant fixed set	Random	
Silva et al. (2021)	Stochastic with Recourse Events are correlated Worst case Distribution	Random	Random No OD parameters as input	-
This work	Stochastic with Recourse Events are correlated Empirical Distribution	Random	Random No OD parameters as input	-

Table 1: Main contributions of crowd shipping routing literature

ODs and what will be the exact expected value of the associated solution by scenario enumeration. The probabilities of acceptance are considered independent. Computational experiments are conducted on randomly generated instances of 15 customers.

In Silva et al. (2021) the authors study a stochastic last-mile delivery with the option of crowd shipping. Here the authors do not consider that uncertain events are independent, and related to customer information only. ODs detailed parameters are not needed to solve the problem. The uncertainty model defined is similar to the one presented in this work, but they assume that the joint probability distribution is difficult to estimate and model it as a data-driven distributionally robust optimization approach to the capacitated vehicle routing problem, where they find an optimal solution for the worst case joint distribution.

For clarity, Table 1 highlights the contributions of each of the previous works, already introducing characteristics of our work for comparison purposes. We defer to Section 3 the details of the approach and contributions of our work.

2.2. Routing with customer uncertainty

One of the first works addressing routing with customer uncertainty was introduced in Jaillet (1988). The author defines a routing problem where only a random subset of customers needs to be visited, following an order previously determined. It is named the Probabilistic Traveling Salesman Problem. Assuming that the probability distribution is known, equal to all customers and independent, the authors derive closed-form expressions for computing efficiently the expected length of any given tour.

In Bertsimas (1992), the authors extend the previous work by considering a probabilistic variant of the classical VRP. In their approach, demands and customer presence are stochastic. They introduce a recourse strategy where absent customers are skipped in the second stage. A major contribution of this work is that the recourse strategy has a potential to perform very closely to a re-optimization strategy, where routes are optimally calculated every time a scenario is released.

Integer L-Shaped branch-and-cut algorithms were proposed in Laporte et al. (1994) and in Gendreau et al. (1995) to solve the two previous models. There the original problem formulation is decomposed into master and subproblems. The authors could solve instances with up to 9 uncertain customers.

A specialized branch-and-bound algorithm is presented in Amar et al. (2017) for the probabilistic travelling salesman problem They adapt existing algorithms for the deterministic traveling salesman problem using the closed expected value evaluation expression defined in Jaillet (1988) and present numerical results for instances up to 18 customers. The same authors present in Amar et al. (2018) another branch-and-bound approach, this time using parallelization techniques, solving instances up to 30 customers.

The authors in Lagos et al. (2017) present an approximation algorithm for the VRP with probabilistic customers. They propose a two-stage stochastic set-partitioning formulation where vehicles are dispatched after observing the subset of customers requiring service; a customer not requiring service is skipped from its planned route at execution, as in Jaillet (1988). For a time limit of six hours, instances up to 40 customers were solved.

A heuristic approach using a VRP set-partitioning formulation with stochastic demand is presented in Novoa et al. (2007). A finite set of feasible routes used as columns to solve the problem is obtained heuristically and an extended recourse strategy is introduced. A planned first-stage route may "fail" when the realized demand at a particular customer exceeds the remaining vehicle capacity. Computational experiments are performed using an instance of 75 customers.

The works presented so far assume that uncertain variables are independent. Nevertheless, the correlations among individual events can contain crucial information. The underlying correlations are often difficult to predict or analyze, which makes the planning problem complicated. This is even more challenging when it considers the large sample size required to characterize joint distribution, since they are potentially high-dimensional.

As an alternative to the independent probability assumption of previous works, the authors in Dinh et al. (2018) and Ghosal and Wiesemann (2018) present a VRP with stochastic demands where there is no recourse, and chance-constrained formulations are used to limit unfeasibility of constraints. The authors propose the use of a sophisticated branch-price-and-cut algorithm. They are examples of modeling using concepts from distributionally robust optimization (DRO), where it is assumed that probability distributions are not completely known and the problem is formulated considering a worst-case probability distribution.

In this work we also assume that uncertain events are correlated, but as

discussed in Section 3, we present a novel approach to solve the problem based on DRL techniques.

2.3. Reinforcement learning for routing

Most works solving VRPs with a RL approach interpret it as a Markov Decision Process. The optimal solution can be viewed as a sequence of actions deciding which customer to visit according to the state revealed. They draw on the concept of policy-gradient or of value-function approximation (VFA). Policy-gradient methods do not maintain a value function, but directly search for an optimal policy. Typically, the policy is parametrized. VFAs approximate the value of post-decision states using simulations. The values are stored in approximation architectures, usually either functions or lookup tables. The challenge is that VRPs can have large combinatorial actions spaces. The resulting high-dimensional action space is impractical for solution approaches that approximate the value of state-action pairs as they typically enumerate all possible actions. Therefore, researchers have avoided searching the entire action space and restricted it instead.

The authors in Bello et al. (2016) present one of the first studies involving RL methods to solve routing problems. They study the deterministic traveling salesman problem (TSP) and train a specialized recurrent neural network, called pointer network, that, given a set of city coordinates, predicts a distribution over different city permutations.

Motivated by the work in Bello et al. (2016), the authors in Nazari et al. (2018) generalize their framework to include a wider range of combinatorial optimization problems such as the VRP. They propose an alternate approach in which the policy model consists of a recurrent neural network (RNN) decoder coupled with an attention mechanism and apply a policy-gradient approach.

An alternative to the approach of reducing the action space with VFA is presented in the work of Delarue et al. (2020). They develop a framework for value-function-based DRL with a combinatorial action space, in which the action selection problem is explicitly formulated as a mixed-integer optimization problem. They focus on the Capacitated Vehicle Routing Problem, where a single capacity-limited vehicle must be assigned one or more routes to satisfy customer demands while minimizing total travel distance. They use neural networks with ReLU activations, leveraging techniques developed in Anderson et al. (2020) to obtain a strong milp reformulation of the action selection problem.

As presented in Section 4, in this work we leverage the idea of Delarue et al. (2020) for the deterministic VRP, extending it to our stochastic problem.

With business models shifting to same-day delivery, routing problems have become increasingly stochastic and dynamic. A problem class called stochastic dynamic vehicle routing problem (SDVRP) arises and poses new challenges as they require anticipatory real-time routing actions and static solutions are no longer adequate. Recent works have shown that RL appears to be a good solution method for dynamic combinatorial optimization as the SDVRP.

Among the papers using RL with NNs applied to dynamic routing problems, Chen et al. (2019b) introduce an actor-critic framework, a policy-based RL

method, for the problem of making pick-ups at customers who make dynamic requests for service. They learn a policy for a single vehicle and then apply this policy to all vehicles.

In Chen et al. (2019a), the authors present among the first papers to implement deep Q-learning techniques for same-day delivery and dynamic routing problems in general. They consider same-day delivery with a heterogeneous fleet of vehicles and drones. They reduce the state space to a set of selected features. The action space also is defined in a way to make it possible to enumerate alternative actions at the decision points.

A survey by Hildebrandt et al. (2021) highlights the potential of RL methods applied to VRPs. Among others, they suggest hybrid approaches that combine piece-wise linear neural network VFAs and strong solvers to search the vast action space under evaluation of future uncertainty. In this work we exploit this suggestion and present a novel policy iteration algorithm as an alternative to previous approaches to handle a SDVRP, as detailed in Section 4.

3. Stochastic crowd shipping last-mile delivery

A typical setting for our problem, but not restricted to it, is a same-day delivery service in which a store serves both as the location for in-store customers and as the depot from where online customer orders are dispatched. In-store customers, who are willing to drop off packages for online customers on their route back home, are potentially offered the service. In return, these in-store customers are offered a small compensation to reimburse their travel costs partially. As the participants are usually free to use any means of transportation to perform the delivery, we refer to them using ODs.

The store provides delivery services throughout fixed time windows during the day. Before each time window, and respecting a process defined by the store, a scenario is revealed with the available, online customer orders and the customers with available ODs. Based on the scenario revealed, the store decides the routes for its fleet of vehicles and which customer orders will be outsourced to ODs. This decision, in turn, defines the cost associated with that time window. The objective is to minimize the total costs in the long run.

The decision is taken in a two-stage approach, using a recourse model based on the work presented in Bertsimas (1992) under the framework of stochastic optimization. An a priori first-stage decision is made during the store planning process, meaning that we define a solution to our problem offline, and before any delivery is initiated. Only the order in which all customers will be delivered is established. The second-stage decision is made every time a scenario is revealed, and before any dispatch of fleet vehicles or ODs. The second-stage decision defines routes that follow the first-stage decision ordering but skips customers that have no online orders and customers outsourced for available ODs. Each time vehicle capacity or the time window limit is reached, a return path to the depot is created and another route restarts from the depot if needed.

The recourse model adopted brings two main advantages to our DRL method presented later in Section 4. First, it extremely reduces the action space since,

in fact, only one decision, defined by the recourse, is possible at each decision point of our model. We recall that RL algorithms typically require an action space that is small enough to enumerate or is continuous. We also note that a very large action space remains to be searched during the first-stage decision, representing possible permutations of customers' delivery ordering. Second, it presents a solution that is potentially very close to the decision adopted in a reoptimization strategy, where an optimal solution is calculated each time a scenario is revealed. The authors in Bertsimas (1992) show that, for their setup where random events associated with customers are assumed independent, both solutions are close, on average. For our case, where random events are not necessarily independent, we will use computational experiments in Section 5 to verify this assumption.

We define two variants for our recourse model, depending on how the ODs are offered the service. In variant 1, if the OD enlists himself/herself to deliver a customer order at the compensation defined, his/her service is immediately accepted even if it is not optimal for the specific scenario being presented. This can be the store's choice because it simplifies the process involved but requires extra care taken in the definition of the OD's compensation. In variant 2, the store only offers the service to the OD if it is optimal for the scenario being revealed.

A vital modeling decision of our approach is that uncertainty is customer-related. We can express not only uncertainty related to a customer with no orders but also uncertainty related to the availability of ODs. It is different from the current crowd shipping last-mile delivery models, where uncertainty is related to the OD (e.g. Dahle et al. (2019); Dayarian and Savelsbergh (2020)). It is suitable for planning purposes and has the advantage that we can reduce the complexity of the problem to be solved by not having to introduce explicit OD's constraints, such as their quantity, capacity and routes, in the problem formulation.

We define a compensation fee to be paid to the OD for each customer. In our model, it pays for only a small detour around each customer. It is equivalent to the idea that the customer will only be crowd shipped if there is an OD with a destination very near him/her. Naturally, there will be a direct relationship between the compensation paid and the availability of ODs. It is compatible with the case where a delivery company would utilize crowd shipping with an emphasis on reducing environmental impacts, like traffic and gas emissions, and not on transforming it into an opportunity for professional services. This compensation strategy is adequate to mitigate the impact of non optimal decisions of variant 1 of our recourse model.

Our approach is also data-driven. For the sake of the scope of this paper, we assume there is a set of historical scenarios available and this set is large enough to train the NN. We notice that in real life that can be not the case. For those cases a data augmentation technique could be applied in order to add newly created synthetic data from existing data (see Shorten and Khoshgoftaar (2019)). In particular, one natural solution to our problem is to exploit the time correlation between the historical data available. Machine learning generative

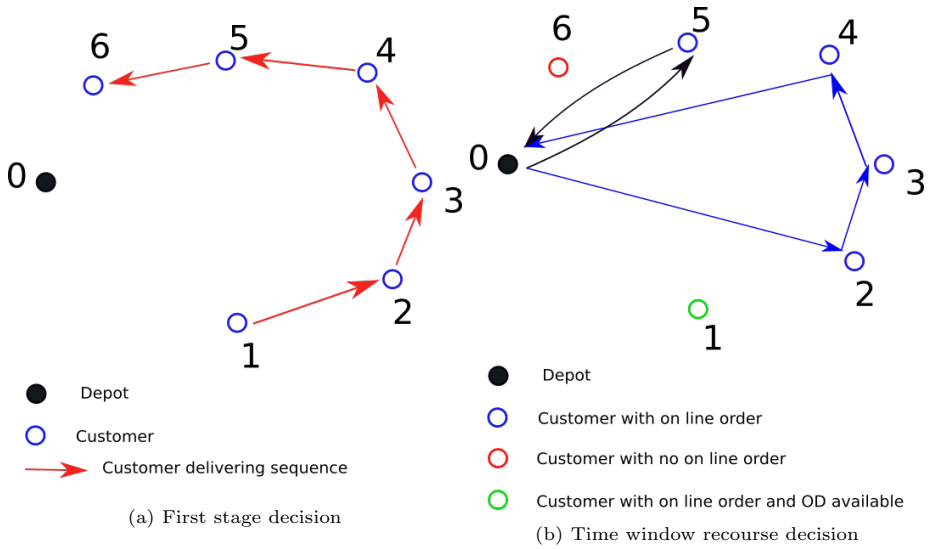
methods (e.g. see conditional generative adversarial networks in Koochali et al. (2021)) can be used to learn the time correlation between scenarios and to artificially generate the additional scenarios needed. We leave this additional algorithm step to another study.

In what follows, we detail the formulation of our problem. Let $G = (V, A)$ be a directed graph, where $V = \{0, \ldots, N\}$ is the set of vertices and $A = \{(i, j)|i, j \in V\}$ is the set of arcs. Set V consists of a depot (vertex 0) and a subset $C = \{1, \ldots, N\}$ of customers' represented by their locations. We assume $|C| \geq 3$ to facilitate our formulations.

A non-negative cost c_{ij} and a duration in time d_{ij} are associated with each arc $(i, j) \in A$. We assume that the graph is symmetric, i.e. $c_{ij} = c_{ji}$; $d_{ij} = d_{ji}$, and that cost and times satisfy triangular inequalities. We also assume that the vehicles to be used as the company fleet are identical and can serve up to Q customers and that all customers to be delivered in a time window must be delivered within a time limit of D. There is a fixed number of K time windows during a day.

The binary vector $\xi = (\xi_1, \ldots, \xi_{2N})^T$ defines a scenario. The vector component $\xi_i = 1$, $i \in \{1, \ldots, N\}$ iff customer i has an online delivery order available and $\xi_i = 1$, $i \in \{N{+}1, \ldots, 2N\}$ iff customer (i-N) has an OD available. If $\xi_i = 0$, $i \in \{1, \ldots, N\}$, customer i will be skipped by the routes defined by the recourse. If $\xi_i = 1$ and $\xi_{i+N} = 1$, $i \in \{1, \ldots, N\}$, customer i delivery order will be considered to be delivered by an OD based on the rules defined for variant 1 and 2 of the recourse model. If $\xi_i = 1$ and $\xi_{i+N} = 0$, $i \in \{1, \ldots, N\}$, customer i delivery order will be served by the fleet of vehicles under routes defined by the recourse. A compensation fee f_i is defined for customer i outsourcing. The support of the joint distribution, Ξ, includes all possible combinations of the scenario's components. We index scenarios using indicator $w \in W = \{1, \ldots, |\Xi|\}$.

Figure 1 exemplifies decisions made and scenarios presented for just 1 time window for a small store setting with only 6 possible customers. We assume here the store has 2 vehicles with a capacity of 3 customer orders each. For this example, we assume there is no need for time delivery constraints. In Figure 1a, circles represent the location of each of the 6 customers in a plane. The depot location is represented by the black circle. There, red arrows represents the first stage decision that defines the order in which the customers will be served after the dispatch of vehicles at each time window. This decision is defined by the sequence of customers 1,2,3,4,5 and 6. Figure 1b represents the 2 vehicle routes in blue and black arrows, defined for the time window after the scenario is revealed. For this time window we define the scenario vector as (1,1,1,1,1,0,1,0,0,0,0,0), meaning that all customers except customer 6 have on line orders to be delivered, and there are ODs available only for customer 1. We assume the decision for the on line order of customer 1 is for it be delivered by an OD. That means that the first vehicle route is defined serving customers 2, 3 and 4 in sequence, and returning to the depot afterwards due to vehicle capacity. The second vehicle route is defined serving only customer 5 and returning to the depot afterwards. Customer 1 is skipped by the vehicles because the respective on line order is delivered by an OD. Customer 6 is skipped because there is no



(a) First stage decision

(b) Time window recourse decision

Figure 1: Store delivery small example

on line order associated. Note that the cost associated with this time window includes not only the cost of the 2 vehicle routes but also the compensation fee paid to the Customer 1 OD.

We model our problem as a Markov decision process (MDP). A MDP models a stochastic and dynamic problem as a sequence of decision points, that is represented by a sequence states connected by actions, defined by policies, rewards and transitions and running through episodes. For our case, a decision point $k \in \{1, \ldots, K\}$ is defined at the beginning of each time window of a day. We recall that decision point is a time at which a decision (recourse action) is made. Overall, at each decision point an action is taken (decision defined by our recourse). This action is taken based on the state that is revealed (scenario presenting what are customers orders and ODs available). The action taken depends also on the first stage decision, z, and, together with the state revealed, incurs in a reward (cost of vehicles routes defined by the recourse plus ODs payment for that time window). A transition then occurs to another decision point (another time window) : this transition is random and defined by a new state (new scenario). The group of K decision points occurs during a day and defines an episode. The sum of all rewards during a episode is defined as total return and is a function of the z. In what follows we give additional details to define the MDP model:

States ξ^k : A state comprises all information needed to select an action and for our problem that is represented by the scenario ξ^k that presents itself right before decision point k.

Actions a^k : Actions a^k implements the recourse model at each decision point and defines routes and ODs allocation. More broadly, it implements our ϵ-

soft policy $\pi \in \Pi$ that is defined by the first-stage decision $z = (z_1, \ldots, z_N)^T \in Z^N$, where $z_i \in \{1, \ldots, N\}$ is the order of delivery of customer i and by the recourse model.

Reward function $R^k(\xi^k, z)$: The reward function evaluates the immediate impact of an action on the objective value. Since each action is a recourse under the defined policy, the reward function is dependent on z and ξ^k. The reward function is defined by the cost of routes plus the ODs payment defined by the recourse. The reward function depends on the recourse model adopted and is detailed in Section 4.

Transitions: Transitions between states are given by exogenous information and related to the time correlation between scenarios. Transitions are defined by the sequence of scenarios available as historical data. We use this sequence of scenarios to perform the Monte Carlo simulation of the Value Function for the MDP model.

Episodes: An episode for our setup problem is a day at the store, composed by K time windows and K decision points. A total return $TR = \sum_{k=1}^{K} R^k(\xi^k, z)$ is defined for each episode.

Value function V: The key idea of RL is the use of value functions to organize and structure the search for good policies. In our problem, each policy π has an expected or mean total return once z is given. The value function V, as a function of z, expresses the expected total return by applying z. If you know the value of each z, then it would be trivial to solve the problem by selecting the ordering of customers z with lowest value.

Objective: A solution to our problem is a policy π that assigns an ordering of customers z and implements a recourse strategy. The optimal solution is a policy π^* that assigns an ordering of customers z^* and minimizes the expected total return and can be expressed by

$$z^* = \underset{z \in Z}{\operatorname{argmin}} \left(V(z) = \mathbb{E}[\sum_{k=1}^{K} R^k(\xi^k, z)]\right) \tag{MVF}$$

4. Deep reinforcement learning for stochastic last-mile delivery with crowd shipping

We implement an on-policy and ϵ-greedy policy iteration algorithm for value-based reinforcement learning with combinatorial actions. We leverage the strategy developed in Delarue et al. (2020) where the authors model the value function as a small NN with a fully-connected hidden layer and rectified linear unit (ReLU) activations. Although it would be natural to use a more complex NN structure since it would lead to a better approximation of the DRL Value function, the authors show that the non-linearity introduced even by a simple NN is sufficient to produce interesting results at the end. We follow here the same design strategy. Other than that, reducing the capacity of the model reduces the likelihood of the model over-fitting the training data set. The capacity of

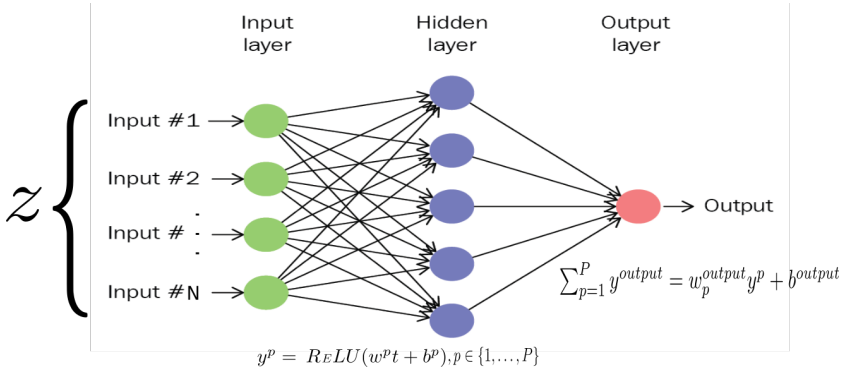


Figure 2: Fully connected neural network with 1 hidden layer and a linear output

a neural network model is defined by both it's structure in terms of nodes and layers and the parameters in terms of its weights. The NN is formulated as a mixed-integer program, as in Anderson et al. (2020), and combined to the combinatorial structure of the action space, the customers delivery ordering, for policy improvement. This, together with the recourse model introduced in our formulation greatly simplifies the complexity of the policy iteration algorithm while maintaining the possibility of searching the entire first-stage decision action space.

Given a randomly chosen starter ϵ-soft policy π_0, where the first-stage decision can vary with probability ϵ, we repeatedly improve it. In the k-th policy evaluation step, using the Monte Carlo method, we repeatedly apply the current ϵ-soft policy π_{k-1} for episodes and average sample total returns after of each episode. The episodes are defined using scenarios provided by historical data. We use the accumulated data generated using the Monte Carlo method to train the NN and incrementally approximate the value function V. We train this NN to minimize the mean-squared error (MSE) on the cumulative cost data gathered among all iterations of our algorithm.

Figure 2 defines the architecture we implement for our NN. The NN has as input the vector z, representing the ordering of customers' deliveries; a single hidden layer, with P hidden nodes, each with a ReLU activation and one linear output. Let $w^p \in R^N$ designate the vector of weights, and $b^p \in R$ the bias term, for the p-th hidden node. Define $w^{output} \in R^P$ and $b^{output} \in R$ analogously for the output layer.

The k-th policy improvement step involves solving the optimization problem related to (MVF). It's solution establishes the first-stage decision to our problem, which minimizes the expected total return expressed by the current approximation of the value function. The minimization problem related to (MVF)

is formulated as (see Anderson et al. (2020))

$$\min \sum_{p=1}^{P} w_p^{output} y^p + b^{output} \tag{1}$$

$$s.t.\, y^p \geq w^p z + b^p \qquad \forall\, 1 \leq p \leq P \tag{2}$$

$$y^p \leq w^p z + b^p + M_-^p (1 - s^p) \qquad \forall\, 1 \leq p \leq P \tag{3}$$

$$y^p \leq M_+^p s^p \qquad \forall\, 1 \leq p \leq P \tag{4}$$

$$y^p \leq \sum_{i \in I} w_i^p (z_i - L_i^p (1 - s^p)) + (b^p + \sum_{i \notin I} w_i^p U_i^p) s^p \qquad \forall\, 1 \leq p \leq P,\, I \subseteq supp(w^p) \tag{5}$$

$$x_{ij} + x_{ji} = 1 \qquad \forall\, i, j \in C,\, i \neq j \tag{6}$$

$$x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad \forall\, i, k, j \in C,\, i \neq j \neq k \tag{7}$$

$$z_i = 1 + \sum_{j \in V,\, j \neq i} x_{ji} \qquad \forall\, i \in C \tag{8}$$

$$z \in R^N,\, y^p \in R,\, s^p \in \{0, 1\},\, x_{ij} \in \{0, 1\} \forall\, i, j \in C,\, i \neq j \tag{9}$$

where the data are:

- $w^p \in \mathcal{R}^N$ designates the vector of weights, and $b^p \in \mathcal{R}$ the bias term, as defined for the p-th hidden node of our neural network.
- In the same way, $w^{output} \in \mathcal{R}^P$ and $b^{output} \in \mathcal{R}$ are defined as for the output layer of our neural network.
- For any vector of weights w, $supp(w)$ indicates the set of indices i such that $w_i \neq 0$
- Components L_i^p and U_i^p are defined as

$$L_i^p = \begin{cases} 0, & \text{if } w_i^p \geq 0. \\ \text{N+1}, & \text{if } w_i^p < 0. \end{cases} \quad \text{and } U_i^p = \begin{cases} \text{N+1}, & \text{if } w_i^p \geq 0. \\ 0, & \text{if } w_i^p < 0. \end{cases}$$

- The formulation's Big-Ms are set as $M_+^p = \max_{z \in [1,N]^N} w^p z + b^p = w^p U^p + b^p$ and $M_-^p = \min_{z \in [1,N]^N} w^p z + b^p = w^p L^p + b^p$.

the variables are:

- We define the N initial decision variables z_i, $1 \leq z_i \leq N$. Variable z_i represents the position in sequence in which customer i will be delivered.
- We define a continuous variable y^p that models the output of the hidden node p.
- The binary variable s^p indicates whether the pre-activation function is positive or negative (i.e. whether the neural network ReLU is active or not).
- We also introduce variables x_{ij} to define the delivery order: $x_{ij} = 1$ if customer i precedes customer j and 0 otherwise.

Objective function (1) minimizes the output of the neural network, representing the output of the value function, and constraints are:

- Constraints (2) to (4) represent a "big-M " linearization of the non-linear output of the ReLU neural network functions. The relationship established by variables s^p is enforced by the "big-M " constraints (3) and (4).
- The exponentially many constraints of type (5) were defined in Anderson et al. (2020) to strengthen the formulation.
- Constraints (6) to (7) define the feasible region of all possible ordering of customers.
- Constraint (8) translates the ordering between customers defined by variables x_{ij} to the equivalent postion in sequence defined by variables z_i.

To be able to solve large instances and still have good solutions, we define a time limit of 1800 s to solve problem (MVF) at each policy improvement step and use the best solution provided until then. We apply warm start, callbacks to introduce lazy constraints and heuristics and use only the needed half of x_{ij} variables, where $i < j$.

We warm start not only in an attempt to accelerate resolution but also to guarantee one incumbent solution. We leverage the study of heuristic approaches for the probabilistic traveling salesman problem in Weiler et al. (2015). In particular, we adapt the Almost Nearest Neighbor Heuristic to our case. Assuming independent marginals, we attempt to find a solution with a maximum lower bound. The ordering of customers is defined by appending the customer with the lowest change of expected length from the last inserted customer to the tour. For a given set T of customers already inserted in a tour, inserting customer j with minimum cost is computed as

$$\min_{j \in C \setminus T} \sum_{i=1}^{|T|} (1 - m_i)(1 - m_j) c_{i,j} \prod_{k=i+1}^{|T|} m_k,$$

where m_i, $i \in C$ is the customer i marginal probability to not be included in a route as defined in the Appendix A for the DRO approach.

Constraints (5) are introduced as cutting planes by lazy constraints callbacks using a linear-time separation routine as described in Anderson et al. (2020). Heuristics callbacks introduce simple heuristics by setting variables x_{ij} as binaries and following the same customer order given by the sort of the z relaxed solution.

Algorithm 1 summarizes the steps undertaken in our policy iteration algorithm. The reward function, $R(.)$, used in algorithm 1 varies if we are using

Algorithm 1 Policy iteration algorithm

Initialize:
  $\epsilon \geq 0$
  $\pi \leftarrow$ an arbitrary ϵ-soft policy π_0 with z_0
$VD \leftarrow \emptyset$ Initialize empty dataset
Repeat for each policy iteration
  Repeat for each episode using scenarios:
    Generate an episode following π: $\xi^0, a^0, R^1, \ldots, \xi^{T-1}, a^{T-1}, R^T$
    $TR \leftarrow 0$
    Loop for each step of episode, $k = T-1, T-2, \ldots, 0$:
      $TR \leftarrow TR + R^{k+1}$
      Append TR to $Returns(z)$
      $VD(z) \leftarrow average(Returns(z))$
  Use VD to incrementally train the NN and approximate value function V
  $z^* \leftarrow \operatorname{argmin} V(z)$ using V function MIP formulation
  Define ϵ-soft policy π with z^*

variant 1 or 2 of the recourse model. Algorithm 2 defines how the reward function is calculated for variant 1 of the recourse model. For variant 1 each ordering position is scanned to verify its customer status defined by the scenario ξ. The action taken is dependent on this status and special care has to be taken to account for the vehicle capacity and the route time duration limit.

We define two forms of reward calculation for variant 2. Here, we want an optimal allocation of ODs. To perform this exactly we formulate this as an

Algorithm 2 Reward function for variant 1 of recourse model

Initialize:
  $laststop = 0$;depot
  $cost = 0$;cost of vehicles route
  $cap = 0$;accumulated capacity of a vehicle
  $time = 0$;accumulated time duration of a vehicle route

for i =1 to N ; scan by delivery ordering positions
  if $\xi[z^{-1}[i]] == 1$ and $\xi[N + z^{-1}[i]] == 0$; customer order and no OD
    if $time + d[laststop, z^{-1}[i]] + d[z^{-1}[i], depot] \leq D$
      $cost+ = c[laststop, z^{-1}[i]]$
      $time+ = d[laststop, z^{-1}[i]]$
      $laststop = z^{-1}[i]$
      $cap+ = 1$
      if $i == N$
        $cost+ = c[laststop, depot]$
      elseif cap == Q
        $cost+ = c[laststop, depot]$
        $laststop = depot$
        $cap = 0$
        $time = 0$
    else
      if $i == N$ # assume 2*time from depot to $i \leq$ D always
        $cost+ = c[laststop, depot] + c[depot, z^{-1}[i]] + c[z^{-1}[i], depot]$
      else
        $cost+ = c[laststop, depot] + c[depot, z^{-1}[i]]$
        $time = d[depot, z^{-1}[i]]$
        $laststop = z^{-1}[i]]$
        $cap = 1$
  elseif $\xi[z^{-1}[i]] == 1$ and $\xi[N + z^{-1}[i]] == 1$; customer order and OD
    $cost+ = f[z^{-1}[i]]$
    if i == N and $cap \neq 0$
      $cost+ = c[laststop, depot]$
  elseif $i == N$ and $cap \neq 0$
    $cost+ = c[laststop, depot]$

optimization problem. The Formulation for this problem is given as

$$\min \sum_{i,j\in V,\, i\neq j} c_{ij}x_{ij} + \sum_{i\in C} f_i w_i$$

$$s.t. \sum_{j\in V,\, i\neq j} x_{ji} = \sum_{j\in V,\, i\neq j} x_{ij} = v_i \qquad \forall i \in C \tag{10}$$

$$\sum_{i\in C} x_{i0} - \sum_{i\in C} x_{0i} = 0 \tag{11}$$

$$v_i + w_i \leq 1 \qquad \forall i \in C \tag{12}$$

$$v_i + w_i \geq \xi_i \qquad \forall i \in C \tag{13}$$

$$w_i \leq \xi_i \qquad \forall i \in C \tag{14}$$

$$v_i \leq \xi_i \qquad \forall i \in C \tag{15}$$

$$w_i \leq \xi_{i+N} \qquad \forall i \in C \tag{16}$$

$$\sum_{j\in V,\, i\neq j} y_{ji} - \sum_{j\in V,\, i\neq j} y_{ij} = v_i \qquad \forall i \in C \tag{17}$$

$$\sum_{j\in C} y_{0j} = \sum_{j\in C} v_j \tag{18}$$

$$y_{i0} = 0 \qquad \forall i \in C \tag{19}$$

$$y_{ij} \leq Q x_{ij} \qquad \forall i, j \in V,\, i \neq j \tag{20}$$

$$\sum_{j\in V,\, i\neq j} t_{ij} - \sum_{j\in V,\, i\neq j} t_{ji} = \sum_{j\in V,\, i\neq j} d_{ij}x_{ij} \qquad \forall i \in C \tag{21}$$

$$t_{0i} \geq d_{0i}x_{0i} \qquad \forall i \in C \tag{22}$$

$$t_{ij} \leq (D - d_{j0})x_{ij} \qquad \forall i, j \in V,\, i \neq j \tag{23}$$

$$\sum_{j\in S} x_{ij} = 0 \qquad \forall i \in C,\, S = \{j |\, z_j < z_i\} \tag{24}$$

$$\sum_{j\in S} x_{ji} \leq v_i \qquad \forall i \in C;\, S = \{j |\, z_j < z_i]\} \tag{25}$$

$$x_{ij} \in \{0,1\}\, \forall i, j \in V,\, i \neq j, y_{ij} \geq 0\, \forall i, j \in V,\, i \neq j \tag{26}$$

$$w_i \in \{0,1\}\, v_i \in \{0,1\}\ \forall i \in C \tag{27}$$

where first-stage decision z and scenario ξ are data input to the problem and we define variables:

- x_{ij} =1 if customer i is served by a vehicle right before j, 0 otherwise,
- w_i =1 if customer i is served by an OD, 0 otherwise,
- v_i =1 if customer i is served by vehicle, 0 otherwise,
- y_{ij} as the accumulated capacity loaded between customer i and j and t_{ij} as the accumulated time spent between customer i and j.

The objective is to minimize total cost of routes plus OD payments, and constraints are:

- Constraints (10) and (11) are route flow conservation and should be considered every time a customer is included in a route, $v_i = 1$.
- Constraints (12) define that customer i is served by vehicle, or an OD or none.
- Constraints (13) define that customer i is served by a vehicle or OD if $\xi_i = 1$.
- Constraints (14) and (15) define that customer i is not served by an OD neither a vehicle if $\xi_i = 0$.
- Constraints (16) define that customer i is served by an OD only if an OD is available.
- Constraints (17) to (20) define the capacity restrictions.
- Constraints (21) to (23) define the time duration restrictions.
- Constraints (24) and (25) guarantee that the order of first-stage decision z is respected.

As an alternative we provide an heuristic for variant 2 calculation where the condition to reduce cost by OD allocation is verified only locally. By Algorithm 3, customers are allocated to ODs only if the cost of paying the OD is less than bypassing until the next available customer using a vehicle route.

5. Experiments and Computational Results

The objective of our experiments is three-fold: we want to analyze the quality of the solution provided by the DRL algorithm. For that we compare to results provided by a reoptimization approach, a sample-average approximation approach and a worst-case optimization approach on small instances. We also compare results from the different recourse variants and compare results of large instances to feasible upper bounds. Additionally, we want to analyze the sensitivity of the algorithm's solution to key parameters configuration. Finally, we want to analyze time performance of the DRL algorithm we have implemented, and with that the capacity to solve large instances.

To pursue this objective, we present in the sections below the instances setting and the implemented benchmark algorithms for the different approaches.

All algorithms are coded in Python, using Keras and Tensorflow, and integrated with Julia (Lubin and Dunning, 2013) using JuMP package and Cplex 12.9. The base code used to run the experiments can be found in `https://github.com/marcostilva/Lastmile_DRL`

We present key parameters and additional architectural features defined for the DRL algorithm and used in the experiment:

Algorithm 3 Reward function for variant 2 of recourse model

Initialize:
  $laststop = 0$;depot
  $cost = 0$;cost of vehicles route
  $cap = 0$;accumulated capacity of a vehicle
  $time = 0$;accumulated time duration of a vehicle route
  $continue = true$; define when to stop algorithm
  $bypass = false$; should bypass OD available
$i = 0$
while continue
  $i+ = 1$
  if $\xi[z^{-1}[i]] == 1$ and ($\xi[N + z^{-1}[i]] == 0$ or $bypass$)
   $bypass = false$
    if $time + d[laststop, z^{-1}[i]] + d[z^{-1}[i], depot] \leq timelimit$
      $cost+ = c[laststop, z^{-1}[i]]$; $time+ = d[laststop, z^{-1}[i]]$
      $laststop = z^{-1}[i]$; $cap+ = 1$
      if $i == N$
        $cost+ = c[z^{-1}[i], depot]$; $continue = false$
      elseif cap == Q
        $cost+ = c[laststop, depot]$; $laststop = depot$; $cap = 0$; $time = 0$
    else
      if $i == N$# assume 2*time from depot to $i \leq$ timelimit always
        $cost+ = c[laststop, depot] + c[depot, z^{-1}[i]] + c[z^{-1}[i], depot]$
        $continue = false$
      else
        $cost+ = c[laststop, depot] + c[depot, z^{-1}[i]]$
        $time = d[depot, z^{-1}[i]]$; $laststop = z^{-1}[i]]$; $cap = 1$
  elseif $\xi[z^{-1}[i]] == 1$ and $\xi[z^{-1}[i] + N] == 1$
   # find next customer available
   $j = i + 1$
    while $j \leq N$ and $\xi[z^{-1}[j]] == 0$)
      $j+ = 1$
    if $j \leq N$ and $f[z^{-1}[i]] \leq c[laststop, z^{-1}[i]] + c[z^{-1}[i], z^{-1}[j]]$
      $cost+ = f[z^{-1}[i]]$
      $i = j - 1$
    elseif $j > N$ and $f[z^{-1}[i]] \leq c[laststop, z^{-1}[i]] + c[z^{-1}[i], depot]$
      $continue = false$
      $cost+ = f[z^{-1}[i]]$
     if $cap \neq 0$
       $cost+ = c[laststop, depot]$
    elseif $j \leq N$ and $f[z^{-1}[i]] > c[laststop, z^{-1}[i]] + c[z^{-1}[i], z^{-1}[j]]$
      $bypass = true$; $i- = 1$
    elseif $j > N$ and $f[z^{-1}[i]] \leq c[laststop, z^{-1}[i]] + c[z^{-1}[i], depot]$
      $bypass = true$; $i- = 1$
  else
    if $i == N$ and $cap \neq 0$
      $cost+ = c[laststop, depot]$
    if $i == N$
      $continue = false$

- We define key parameters with default values: number of nodes of hidden layer of the NN as 16, number of policy iterations as 15 and number of training episodes as 300000. For some of the experiments, when specified, we change default values to analyze the sensitivity of the DRL method to these changes.
- Exploration and exploitation during training is performed by setting ϵ-soft policies. We set the probability of exploring $\epsilon = 0.6$ and exploiting $1 - \epsilon$ and decay ϵ over the policy iterations.
- For weight updates, we use a learning rate that exponentially decays from 0.01 with the base 0.96 and the decay rate 1/6000.
- We pass through the entire episodes dataset 100 times (epochs) with a batch size of 100.

5.1. Instances

We generate random test instances having $|C| + 1$ vertices (depot and $|C|$ customers) for different values of $|C| \in \{10, 15, 18, 30, 40, 50, 70, 150, 300\}$. Five instances for each number of customers are generated. All results presented by the number of customers are an average of all of their respective instances.

Customers' locations for each instance are assigned randomly from a grid of 100*100 possible locations. We assume that travel time (in seconds) and cost (in monetary value) are deterministic and proportional to the euclidean distances between customers. We do that to simplify instance data generation only and it is not a requirement for our algorithm.

The compensation fee f_i for each customer i is set to a fixed small value, to avoid zero compensation fees, plus a value proportional to the minimal detour considering all pairs of customers $r, j \in C, i \neq j \neq r$ and given by $f_i = \min_{j,r \in C} c_{j,i} + c_{i,r} - c_{j,r}$.

Customers' orders and OD availability occur randomly around the day and present themselves for each time window as scenarios. We assume there is a set of scenarios available as data and that is sufficient to train the NN. We artificially generate these scenarios for our test instances based on two probability vectors that define marginal probabilities for customers' orders (m_i) and OD availability for each customer (o_i). To assure scenario consistency, the OD availability is only assigned when the respective customer is also assigned to a delivery order. To introduce correlation between scenarios we force customers to have a maximum of 1 delivery order per day. We generate 800000 scenarios that are used to train the NN plus 1500 scenarios that are used to simulate the solutions provided by the algorithms (out-of-sample performance). Note that we validate the solutions provided by the different algorithms, through simulation , using scenarios different from the ones used for training our neural network. This way we can identify potential overfitting raised during training. Note also that the use of large data sets to train our neural network is a good tool to mitigate overfitting.

Algorithm Code	Description
$DRLV1$	The DRL method with recourse variant 1
$DRLV2E$	The DRL method with recourse variant 2 exact formulation
$DRLV2H$	The DRL method with recourse variant 2 heuristic
SAA	The sample-average approximation method with recourse variant 1
$DROA$	DRO algorithm with recourse variant 1
$REOPT$	Optimal routes calculation for each scenario (on line decision). OD paid if available

Table 2: Algorithms

We assume that the pairs (o_i, f_i) generated are coherent, in the sense that the compensation fee influences the probability of an OD accepting to outsource.

The values m_i and o_i are assigned randomly for each instance in a range smaller than 0.3.

Each episode is composed of a delivery day with 4 time windows of 2 hours each, and therefore, 4 scenarios.

The professional fleet vehicle capacity is set to $Q = \left\lfloor \frac{|C|}{3} \right\rfloor$ and the time limit of a route is given by the time windows of 2 hours.

5.2. Benchmark algorithms

We present in Table 2 a general description of the different algorithms we run our instances with.

Besides implementing algorithms $DRLV1$, $DRLV2E$ and $DRLV2H$ for the methods presented in Section 4, we implement algorithms SAA, $DROA$ and $REOPT$ to run the same instances.

A common approach to solve a stochastic problem is to extract a sample of the uncertain parameters and model the expected performance using the sample average. Optimizing the sample average yields the sample-average approximation (SAA) algorithm of stochastic programming (Kleywegt et al. (2002)). For a given random sample of size m episodes from the available historical data, the SAA estimate of the performance of a given delivery order, z, is given by

$$\hat{V}_m(z) = \sum_{i=1}^{m} \sum_{k=1}^{K} R^k(\xi^{i,k}, z)$$

where we index each scenario ξ by the episode i and stage k. We have set $m = 50$ to run our experiments.

The SAA approach is justified by the fact that if the sample is drawn i.i.d, then $\hat{V}_m(z)$ is an unbiased estimate of $V(z)$ and by the strong law of large numbers it will converge almost surely to $V(z)$. The SAA approach aims to optimize the true performance $V(z)$ by optimizing the SAA estimate of performance:

$$\min_{z \in Z} \hat{V}_m(z)$$

The solution value of SAA is the in-sample performance, i.e., the average performance over the sample of uncertain m episodes. What is more important is the performance of the solution provided out-of- sample, i.e., under the true

distribution of episodes. We formulate SAA as a MILP optimization problem and it is tractable for only small values of the sample size.

Algorithm $DROA$ is a DRO implementation approach. DRO has emerged from within the optimization community as an approach that explicitly accounts for the fact that one is never able to exactly specify a probability distribution in practice. DRO weakens the requirement to specify a single probability distribution for the uncertain parameters. Instead, a set of possible probability distributions is defined and the problem is optimized for the worst-case distribution within this set. The $DROA$ algorithm implemented is presented in detail in Silva et al. (2021). It implements an exact two-stage distributionally robust optimization approach with the variant 1 recourse. It searches an optimal first-stage solution for the worst-case scenario distribution, where the set of feasible scenario probability distributions is defined by the marginal distributions of the scenarios that we calculate from the historical data availabe for each problem instance. We impose a time limit of 4 hours to run $DROA$. In case the time limit is reached, we use the best solution provided so far by the algorithm. We run $DROA$ for the smaller instances only. For completeness, we provide more information on the algorithm in the Appendix A.

Algorithm $REOPT$ solves a MILP formulation implementing the reoptimization strategy, where an optimal route is calculated for each scenario (there is no first-stage solution). It is equivalent to implementing on line solutions. We run $REOPT$ for the smaller instances only.

5.3. Solution Quality

To assess the performance of the solutions provided by the different algorithms, we simulate these solutions through various episodes using the scenarios created for this purpose, providing an out-of-sample estimate. We compare the algorithms total cost output of this simulation. To have good solutions from SAA, $DROA$, and $REOPT$ algorithms we run this comparison only for instances where $|C| \in \{10, 15, 18, 30, 40\}$. We run $DRLV2E$ out-of-sample simulation with the solution from $DRLV2H$ for instances where $|C| \geq 30$.

Table 3 reports, for the subset of instances, the average percentage gap between total cost values when compared to the $REOPT$ algorithm total cost.

Overall, we observe that all algorithms provide total costs within a range of 20% of the $REOPT$ total cost for the simulation proposed. The cost gap increases for larger $|C|$. That is in part explained by the parameters configuration used to define solutions as will be better analyzed in Section 5.4. Here we advance that the solution quality of the DRL approach is directly related to the number of episodes used for training, and as the number of customers increases so does the need to increase the number of training episodes.

$DRLV2E$ provides a simulated total cost that is always smaller than $DRLV1$. That is explained because of the extra freedom that recourse variant 2 has in $DRLV2E$ to accept or not the OD. It comes as a bit of surprise, though, given the method adopted to calculate the OD's compensation fee to mitigate sub-optimization. On the other hand, $DRLV2H$ is not able to improve total costs

over $DRLV1$ exactly because of the method adopted to calculate compensation fees. We note that could not be the case if, for instance, the compensation fees were increased. To experiment that we run the same instances, but increasing the OD's compensation fee by a factor of 6 for the same OD's marginal probability. Table 4 presents the results. There is an average of 2.2% decrease on savings compared to $DRLV1$ total costs when adopting recourse variant 2 with heuristic ($DRLV2E$).

We also present in Table 5 the average percentage of ODs not accepted to outsource a customer, among those available. We can see that, although the compensation fee strategy implemented to mitigate sub-optimization, there is room for improvement by adopting the exact variant 2 approach.

$DROA$, although an exact approach, is not able to improve results given by $DRLV1$. There is a caveat here: by the design of our experiment training and out-of-sample simulation scenarios are all drawn from the same distribution. Since $DROA$ is the best solution for a worst-case probability distribution, the better results of $DRLV1$ would have been expected if the $DRLV1$ algorithm had been able to correctly learn the scenario probability distribution. By the results shown in Section 5.4, we see that the quality of $DRLV1$ results are related to the number of scenarios used for training. We then experiment comparing $DROA$ results to $DRLV1$ for different number of training scenarios. Table 6 presents solution quality for different number of training scenarios. We see that $DROA$ provides better results for small number of scenarios. This suggests that the strength of our algorithm $DRLV1$ is in the fact it can process large number of scenarios and, by doing this, provide good quality solutions even for large instances. Our problem is suitable for $DRLV1$ because of the recourse model adopted that is very fast to solve at each stage of the MDP process. Additionally, it is adequate to provide a large number of scenarios. The scenarios can be generated artificially by exploiting the time correlation between scenarios, for instance. We note that the $DROA$ approach would be more suitable in the case the scenarios probability distribution were only partially known and, for instance, only a small number of scenarios were available during planning (definition of delivery order) phase. The $DROA$ approach reduces the variance in the resulting simulations.

The SAA results in Table 3 presents higher variance when compared to other algorithms. In other words these results show that with a small sample size the SAA approach is susceptible to over-fitting, motivating the question of whether an alternative methodology can produce designs that exhibit better out-of-sample performance when provided with the same data.

To verify the quality of the solution of larger instances, we first estimate an upper bound by running the out-of-sample simulation with a random generated customer ordering as input. Table 7 presents the results as a percentage gap between the upper-bound cost ($UPPERBOUND$) and $DRLV1$ cost. There is an average improvement of 19,45% by running $DRLV1$ solutions when compared to $UPPERBOUND$.

$\|C\|$	$DRLV1^*$	$DRLV2E^*$	$DRLV2H^*$	$DROA^*$	SAA^*
10	5.4	5.2	5.4	5.4	10.5
15	6.5	6.1	6.5	6.1	8.4
18	8.2	6.8	8.2	10.7	15.1
30	11.5	9.3	11.5	15.6	17.3
40	15.3	12.7	15.3	19.7	14.3

Table 3: Solution Quality
* results presented as percentage gap when compared to $REOPT$.
Result = $100 * AVG((algorithm - REOPT)/REOPT)$

$\|C\|$	$DRLV2H$
10	-2.7
15	-3.5
18	-1.6
30	-1.3
40	-2.0

Table 4: Solution Quality for increased compensation fees
* results presented as percentage gap when compared to $DRLV1$.
Result = $100 * AVG((DRLV2H - DRLV1)/DRLV1)$

$\|C\|$	$DRLV2E$
10	4.7
15	5.3
18	3.8
30	7.5
40	8.3

Table 5: Percentage of ODs not accepted for $DRLV2E$

Scenarios	$DRLV1$	$DROA$
10000	36.5	11.5
30000	24.6	11.5
300000	9.3	11.5
500000	8.1	11.5
800000	8.1	11.5

Table 6: Solution Quality for different number of training scenarios

$\|C\|$	Gap (%)*
50	-15.7
70	-12.8
150	-26.5
300	-22.8

Table 7: $DRLV1$ Solution Quality for larger instances
* results presented as percentage gap when compared to $UPPERBOUND$.
Result $= 100 * AVG((DRLV1 - UPPERBOUND)/DRLV1)$

5.4. Sensitivity to parameters configuration

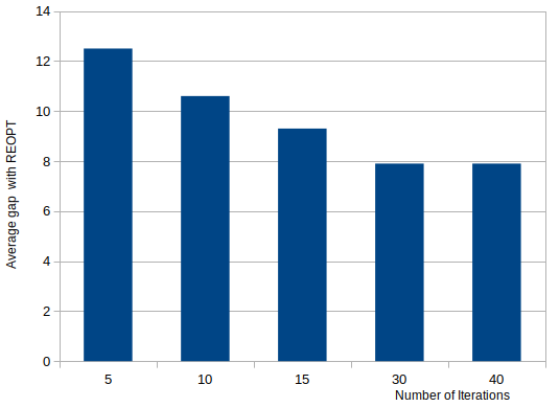
In this section, we analyze the effect of changing the number of policy iterations, the number of training scenarios, and the number of the NN hidden layer nodes on the solution quality, presented as the percentage average gap between the total cost output of the simulation running recourse variant 1 with the $DRLV1$ first-stage solution, to the $REOPT$ total cost output of the simulation. We change each of the parameters independently while maintaining the other parameters as default. This is reported in Figures 3a, 3b and 3c, respectively.

By Figures 3a and 3b, we note that decreasing the number of training scenarios can be compensated by increasing the number of policy iterations to maintain solution quality and vice versa. It would be a matter of identifying which combination of both provides best time performance. Since the policy evaluation phase of our algorithm is very fast, due to the simple recourse, we have opted to increase the number of training scenarios as default.

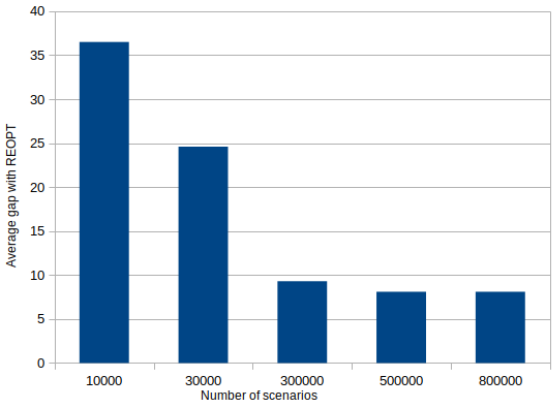
In Figure 3c we analyze the effects of increasing NN size on the solution quality. We experience the same effect as with the other experiments. Overall, the number of nodes is a determinant of the solution quality.

5.5. Algorithms performance

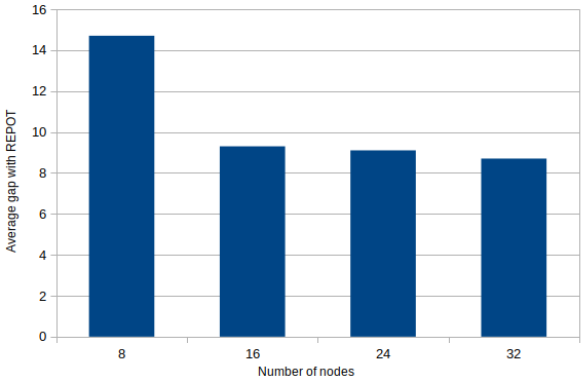
In Table 8 we present the time performance of algorithms $DRLV1$ and $DROA$. It reports the average total time to find offline the first-stage solution, by $|C|$. We run $DROA$ only for instances where $|C| \leq 18$ because after that only a small percentage of instances are solved under the time limit. We see that different from $DRLV1$, $DROA$ has time performance that is exponential in the number of customers, $|C|$. $DROA$ only performs better than $DRLV1$ for small instances. On the other hand, the time spent by $DRLV1$ can be altered by adjusting the solution quality using parameters number of policy iterations, number of training episodes, and number of NN nodes. $DRLV1$ scales well for the number of customers and can be used to solve large instances. Since both algorithms run offline, they can be used for dynamic decisions when the scenarios are revealed. From Table 8 we infer also that $DRLV1$ should be preferred when there are enough historical scenarios to train the NN.



(a) Effect of number of policy iterations



(b) Effect of number of training scenarios



(c) Effect of number of NN hidden layer nodes

Figure 3: Sensitivity to key parameters

$\|C\|$	$DRLV1$	$DROA$
10	874.26	21.6
15	1528.00	3247.81
18	1746.32	18526.76
30	3010.11	-
40	3411.87	-
50	4892.66	-
70	8914.00	-
150	22650.06	-
300	51457.79	-

Table 8: Time performance in seconds

6. Conclusion

We present a novel solution approach for the stochastic and dynamic crowd shipping last-mile delivery problem and solve it approximately using a DRL method. In our approach, it is possible to capture uncertainty related to customers' online orders and occasional drivers' availability. The integration of machine learning and operations research optimization techniques have worked as an appropriate alternative to handle the large state and action space.

Computational results demonstrate that the method is capable of making appropriate decisions throughout the day resulting in total costs that approximate a reoptimization approach, where optimal routes are calculated every time a scenario is revealed.

Overall, we compare different implemented algorithms' solutions and performance. We analyze different recourse strategies for the DRL method and compare to exact approach solutions, not only the reoptimization approach, but also a worst-case distribution approach where we assume only partial information of the scenario probability distribution is known.

We foresee directions for future research. The challenge of solving larger instances can motivate future development of algorithmic methods using a more sophisticated DRL approach, for instance. Many steps of the algorithms can be performed in parallel mode. Action-value learning algorithms, instead of a value-based function approximation approach as we have implemented can be studied. Generative machine learning methods can be studied to satisfy the need for a large amount of sampled scenario data. Different neural network architectures, that better capture the sequence dependence nature of the problem, can be used to better approximate the DRL method value function.

References

Alnaggar, A., Gzara, F., Bookbinder, J.H., 2019. Crowdsourced delivery: A review of platforms and academic literature. Omega , 102139.

Amar, M.A., Khaznaji, W., Bellalouna, M., 2017. An exact resolution for the probabilistic traveling salesman problem under the a priori strategy. Procedia Computer Science 108, 1414 – 1423. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

Amar, M.A., Khaznaji, W., Bellalouna, M., 2018. A parallel branch and bound algorithm for the probabilistic tsp, in: Vaidya, J., Li, J. (Eds.), Algorithms and Architectures for Parallel Processing, Springer International Publishing, Cham. pp. 437–448.

Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.P., 2020. Strong mixed-integer programming formulations for trained neural networks. Math. Program. 183, 3–39.

ARC Advisory Group, 2021. What are omni-channel fulfillment and returns management all about? URL: `\tiny{https://www.arcweb.com/industry-best-practices/what-omni-channel-fulfillment-returns-management-all-about}`. last access on 2021-10-18.

Archetti, C., Guerriero, F., Macrina, G., 2021. The online vehicle routing problem with occasional drivers. Comput. Oper. Res. 127, 105144.

Archetti, C., Savelsbergh, M.W.P., Speranza, M.G., 2016. The vehicle routing problem with occasional drivers. European Journal of Operational Research 254, 472–480.

Arslan, A.M., Agatz, N., Kroon, L., Zuidwijk, R., 2019. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. Transportation Science 53, 222–235.

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. CoRR abs/1611.09940. URL: `http://arxiv.org/abs/1611.09940`.

Bertsimas, D.J., 1992. A vehicle routing problem with stochastic demand. Operations Research 40, 574–585.

Chen, X., Ulmer, M.W., Thomas, B.W., 2019a. Deep q-learning for same-day delivery with a heterogeneous fleet of vehicles and drones. CoRR abs/1910.11901. URL: `http://arxiv.org/abs/1910.11901`.

Chen, Y., Qian, Y., Yao, Y., Wu, Z., Li, R., Zhou, Y., Hu, H., Xu, Y., 2019b. Can sophisticated dispatching strategy acquired by reinforcement learning? - a case study in dynamic courier dispatching system. `arXiv:1903.02716`.

Dahle, L., Andersson, H., Christiansen, M., 2017. The vehicle routing problem with dynamic occasional drivers, in: Bektaş, T., Coniglio, S., Martinez-Sykora, A., Voß, S. (Eds.), Computational Logistics, Springer International Publishing, Cham. pp. 49–63.

Dahle, L., Andersson, H., Christiansen, M., Speranza, M.G., 2019. The pickup and delivery problem with time windows and occasional drivers. Computers & OR 109, 122–133.

Dayarian, I., Savelsbergh, M., 2020. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. Production and Operations Management 29, 2153–2174.

Delarue, A., Anderson, R., Tjandraatmadja, C., 2020. Reinforcement learning with combinatorial actions: An application to vehicle routing, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Dinh, T., Fukasawa, R., Luedtke, J., 2018. Exact algorithms for the chance-constrained vehicle routing problem. Mathematical Programming 172, 105–138.

Doordash, 2021. Delivering with Doordash. URL: `https://www.doordash.com/about/`. last accessed on 2021-10-18.

Gdowska, K., Viana, A., Pedroso, J.P., 2018. Stochastic last-mile delivery with crowdshipping. Transportation Research Procedia 30, 90 – 100.

Gendreau, M., Laporte, G., Séguin, R., 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science 29, 143–155.

Ghosal, S.K., Wiesemann, W., 2018. The distributionally robust chance constrained vehicle routing problem. available in "http://www.optimization-online.org/DB_FILE/2018/08/6759.pdf" .

Hildebrandt, F.D., Thomas, B.W., Ulmer, M.W., 2021. Where the action is: Let's make reinforcement learning for stochastic dynamic vehicle routing problems work! CoRR abs/2103.00507. URL: `https://arxiv.org/abs/2103.00507`.

Huang, K., Ardiansyah, M.N., 2019. A decision model for last-mile delivery planning with crowdsourcing integration. Computers & Industrial Engineering 135, 898–912.

Jaillet, P., 1988. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. Operations Research 36, 929–936.

JD-Dada, 2021. Become a Dada Knight. URL: https://www.imdada.cn/. last accessed on 2021-10-18.

Kafle, K., Zou, B., Lin, J., 2017. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. Transportation Research Part B: Methodological 99, 62–82.

Kleywegt, A.J., Shapiro, A., Homem-de Mello, T., 2002. The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization 12, 479–502.

Koochali, A., Dengel, A., Ahmed, S., 2021. If you like it, gan it—probabilistic multivariate times series forecast with gan. Engineering Proceedings 5.

Lagos, F., Klapp, M., Toriello, A., 2017. Branch-and-price for probabilistic vehicle routing. available in "http://www.optimization-online.org/DB_HTML/2017/12/6364.html" .

Laporte, G., Louveaux, F.V., Mercure, H., 1994. A priori optimization of the probabilistic traveling salesman problem. Operations Research 42, 543–549.

Lubin, M., Dunning, I., 2013. Computing in Operations Research using Julia. CoRR abs 1312.1431.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laganà, D., 2017. The vehicle routing problem with occasional drivers and time windows, in: Sforza, A., Sterle, C. (Eds.), Optimization and Decision Science: Methodologies and Applications, Springer International Publishing. pp. 577–587.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laporte, G., 2020. Crowd-shipping with time windows and transshipment nodes. Computers & Operations Research 113.

Nazari, M., Oroojlooy, A., Snyder, L.V., Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem, in: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 9861–9871.

Novoa, C., Berger, R., Linderoth, J., Storer, R., 2007. A set-partitioning-based model for the stochastic vehicle routing problem. available in "http://www.optimization-online.org/DB_HTML/2006/12/1542.html" .

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. Journal of Big Data 6.

Silva, M., Pedroso, J.a.P., Viana, A., Klimentova, X., 2021. A Branch-Price-And-Cut Algorithm for Stochastic Crowd Shipping Last-Mile Delivery with

Correlated Marginals, in: Müller-Hannemann, M., Perea, F. (Eds.), 21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany. pp. 12:1–12:20.

Walmart, 2021. Spark Driver Delivery. URL: `https://drive4spark.walmart.com/`. last accessed on 2021-10-18.

Weiler, C., Biesinger, B., Hu, B., Raidl, G.R., 2015. Heuristic approaches for the probabilistic traveling salesman problem, in: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (Eds.), Computer Aided Systems Theory – EUROCAST 2015, Springer International Publishing, Cham. pp. 342–349.

Acknowledgment

This work is partially funded by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project POCI-01-0145-FEDER-028611.

Appendix A. Distributionally robust optimization approach to last-mile delivery with crowdshipping

In the *DROA* approach, after defining a set $\mathcal{P}$ of feasible probability distributions that is assumed to include the true distribution, the objective function is reformulated with respect to the worst-case expected cost over the choice of a distribution in this set. This leads to solving the Distributionally Robust Optimization Problem

$$\min_{z \in Z} \max_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}[R(z, \xi)],$$

where $R(z, \xi)$ is a reward function in z, first-stage solution, that depends on the vector of random parameters ξ, and $\mathbb{E}_{\mathbb{P}}$ is the expectation taken with respect to the random vector ξ given that it follows the probability distribution $\mathbb{P}$. The set $\mathcal{P}$ is called the ambiguity set.

Since an ambiguity set only characterizes certain properties of the unknown true probability distribution, its estimation requires fewer data and can often be done using historical records, being suitable for data-driven approaches.

In the *DROA* approach we simplify the definition of the scenario vector, when compared to the *DRL* approach, to reduce complexity of the *DROA* algorithm. Here, vector $\xi = (\xi_1, \ldots, \xi_N)$ defines an uncertain scenario, $\xi_i = 1$ iff $i \in C$ is skipped, 0 otherwise. A customer is skipped, meaning it will not be part of any vehicle route, if there is no online order for that customer or if there is an OD available to deliver his/her order. The support of the joint

distribution, Ξ, includes all possible combinations of the scenario's components. We index scenarios using indicator $w \in W$. For each scenario with customer i being skipped there is a marginal probability, m_i, and a compensation fee, f_i, associated. We compute the marginal probability m_i from the set of available historical data. As a remark, note that f_i is the compensation fee paid to the OD, weighted by the probability of the customer being outsourced. We assume that the uncertain components are not independent and the joint distribution is unknown.

We define our ambiguity set as

$$\mathcal{P} = \{\mathbb{P} \,|\, \mathbb{P}\{\xi \in \Xi\} = 1; \text{ marginals } m_i \text{ for } \xi_i = 1,\, i \in C\},$$

Using the definition of the ambiguity set we dualize the inner maximization problem of the $DROA$ formulation and arrive to the following reformulation

$$\begin{aligned}
\min_{z \in Z} \quad & s - \sum_{i \in C} m_i u_i & \\
s.t. \quad & s - \sum_{i \in C} \xi_i^w u_i \geq R(z, w) & \forall w \in W \\
& s \geq 0,\, u_i \geq 0 & \forall i \in C
\end{aligned}$$

where s, u_i, $i \in C$ are dual variables introduced in the reformulation.

We define the first and second stage formulations, including the reward function $R(z, \xi)$. The first stage is defined solely by a ordering for serving the customers. The following variables are used:

- First-stage main variable

  $z_{i,j} = 1$ iff customer i is served before customer j.

- First-stage auxiliary variables

  $z^1_{i,j,r} = 1$ iff customer r is served in between customers i and j
  $z^2_{i,j,r} = 1$ iff customer r is served before customers i and j
  $z^3_{i,j,r} = 1$ iff customer r is served after customers i and j

The second stage is defined in a way that we can calculate the cost of a route given the ordering of the first stage and the scenario to be considered. We define the following sets of main and auxiliary second-stage variables, where now we include the depot in the ordering as it will be always the first and last to be served in each route:

- Main variables

  $y_{w,i,j} = 1$ iff, for scenario ξ^w, depot or customer j is served right after depot or customer i. This means that all customers r in between i and j are outsourced in this scenario.

$v_{w,i,j} = 1$ iff, for scenario ξ^w, vehicle capacity, Q, is reached at customer i and j is the next not skipped customer. This means that before customer i, in scenario ξ^w, there are $kQ-1$ customers, where $k \in \{1, \ldots, \left\lfloor \frac{|C|}{Q} \right\rfloor\}$ and that all customers r in between i and j are outsourced in this scenario.

- Auxiliary variables

  $y^1_{w,i,t} = 1$ iff, for scenario ξ^w and given a ordering of customers, there are t customers before i, $t \in \{0, \ldots, |C|-1\}$. It indicates the position of a customer for each scenario.

The reward function sums up the cost of each arc transpassed considering all routes plus the cost of the outsourced customers. We have already stated that each variable $y_{w,i,j} = 1$ defines an arc that is transpassed and each variable $v_{w,i,j} = 1$ defines a detour to the depot. This way we define the reward function as

$$R(z,w) = \sum_{i \in C} f_i \xi_i^w + \sum_{\substack{i,j \in V \\ i \neq j}} c_{i,j} y_{w,i,j} + \sum_{\substack{i,j \in C \\ i \neq j}} (c_{i,0} + c_{0,j} - c_{i,j}) v_{w,i,j},$$

With all variables and reward function defined we formulate the *DROA*

approach as

$$
\begin{aligned}
\min \quad & s + \sum_{i \in C} m_i u_i \\
s.t. \quad & s + \sum_{i \in C} u_i \xi_i^w \geq \sum_{i \in C} f_i \xi_i^w + \sum_{i,j \in V} c_{i,j} y_{w,i,j} + \sum_{i,j \in C} (c_{i,0} + c_{0,j} - c_{i,j}) v_{w,i,j} \\
& z_{i,j} + z_{j,i} = 1 \\
& z_{i,j} + z_{j,r} + z_{r,i} \leq 2 \\
& z^1_{i,j,r} \geq z_{i,r} + z_{r,j} - 1 \\
& z^2_{i,j,r} \geq z_{r,i} + z_{r,j} - 1 \\
& z^3_{i,j,r} \geq z_{i,r} + z_{j,r} - 1 \\
& y_{w,i,j} \geq 1 - \xi_i^w + 1 - \xi_j^w + z_{i,j} + \sum_{r \in C} (\xi_r^w z^1_{i,j,r} + z^2_{i,j,r} + z^3_{i,j,r}) - |C| \\
& y_{w,0,i} \geq 1 - \xi_i^w + \sum_{j \in C} (\xi_j^w z_{j,i} + z_{i,j}) - |C| + 1 \\
& y_{w,i,0} \geq 1 - \xi_i^w + \sum_{j \in C} (\xi_j^w z_{i,j} + z_{j,i}) - |C| + 1 \\
& v_{w,i,j} \geq y_{w,i,j} + \sum_{k \in \{1,\ldots,\left\lfloor \frac{|C|}{Q} \right\rfloor\}} y^1_{w,i,kQ-1} - 1 \\
& \sum_{t \in \{0,\ldots,|C|-1\}} y^1_{w,i,t} = 1 - \xi_i^w \\
& \sum_{t \in \{0,\ldots,|C|-1\}} t y^1_{w,i,t} \leq \sum_{j \in C} (1 - \xi_j^w) z_{j,i} \\
& \sum_{i \in C} y^1_{w,i,t} \leq 1 \\
& s \geq 0,\ u_i \leq 0 \\
& z^1_{i,j,r},\ z^2_{i,j,r},\ z^3_{i,j,r} \in [0,1],\ z_{i,j} \in \{0,1\} \\
& y^1_{w,i,t}, y_{w,i,j},\ y_{w,0,i},\ , y_{w,i,0},\ v_{w,i,j} \in [0,1]
\end{aligned}
$$

where the constraints and variables are valid $\forall w \in W$, $\forall i,j,r \in C$, $i \neq j \neq r$, and $\forall t \in \{0,\ldots,|C|-1\}$, when not stated otherwise.

To solve the above formulation we propose a branch-price-and-cut algorithm.

Algorithm 4 summarizes the main steps undertaken to perform the branch-price-and-cut algorithm. The directives of the implementation of the algorithm are:

- A customized branching rule based on the incremental ordering of the sequence of the visit of the customers. This branching rule permits that we fix many binary variables simultaneously to their lower or upper bounds at a node while producing feasible regions of equitable sizes after branching.
- A symmetry breaking strategy to limit the number of branchings. This is a way to eliminate partial orderings of customers that will not contribute to arriving at an optimal solution and therefore gain greater computational efficiency by eliminating nodes of our branching tree.

- At each node solve a relaxed restricted version of the formulation. The restricted version is composed of a finite number of scenarios.
- Initial tests indicate that the node relaxation is weak and may consume significant time. On the other hand, the independent marginal distribution version of the formulation provides a lower bound that is easy to calculate at each node. We then use this alternative as a lower bound to prune the nodes before proceeding with the calculation of the relaxed restricted version of our problem.
- Each node is solved to optimality and is pruned by its lower bound.
- Each node's integer solution is validated against new scenarios. A separation subproblem with a column and row generation approach is used to separate invalid integer solutions.
- New scenarios inserted re-initiate the process of solving the node relaxed problem.
- Valid integer solutions are tested against the incumbent solution and the correspondent node is pruned afterwards.
- Fractional solutions are branched.
- The algorithm runs until no more nodes are available to test or when a time limit is reached

Algorithm 4 Branch-price-and-cut (BPC) algorithm

Input ▷ Q, set C, vectors c, f, m
Initialize
//Nodes list ← root node, Incumbent solution ← Heuristic, Lower bound ← $-\infty$
while There are still nodes to be branched in the Nodes list **do**
    Node Select ▷ Select node based on search criteria
    Initialize scenarios ▷ Add scenarios from parents node
    Prune ▷ by Independent lower bound
    while There are still scenarios to be added **do**
        Solve
        Prune ▷ by Node solution-lower bound
        Scenario Separation subproblem ▷ If integer
    end while
    Update if new Incumbent solution ▷ Prune if better value
    Branch node
    Prune ▷ by symmetry
    Update Nodes List
end while
Return optimal solution - order of customers to visit and expected cost