Omega 140 (2026) 103478

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Dynamic stochastic parcel locker assignment with uncertain pick-up times☆

Simona Mancini [a,b,*], Margaretha Gansterer [b]

[a] University of Palermo, Department of Engineering, Palermo, Italy
[b] University of Klagenfurt, Department of Operations, Energy, and Environmental Management, Universitaetsstraße 65-67, 9020 Klagenfurt, Austria

ARTICLE INFO

Keywords:
Last-mile delivery
Stochastic optimization
Parcel lockers

ABSTRACT

Automated parcel lockers are used by logistics providers in order to increase the efficiency of last-mile delivery operations particularly in urban areas. We consider the case of a company that operates lockers and dynamically receives delivery orders that have to be accepted or rejected immediately. On a second decision stage, the set of accepted orders has be assigned to the lockers such that customer compatibility requirements and maximum fulfillment times are respected. As both future arrivals of orders as well as customer pickup-times are unknown, the company faces a dynamic stochastic problem. To generate solutions, we propose a decision framework based on a classification approach. The classifier uses a mixed integer model to learn from optimal solutions within a deterministic setting and exploits this information within the dynamic stochastic process. We assess the proposed method within an extensive computational study where both artificial instances and a real world case are addressed. The obtained results show that the classification-based framework outperforms all benchmark methods, which include (i) scenario sampling, (ii) classical decision trees, and (iii) several deterministic policies. Managerial insights with regard to most important systems' features within the decision process are derived. The newly proposed decision framework is generalizable such that it can be applied to related dynamic stochastic matching problems.

1. Introduction

All figures draw the same picture: e-commerce is here to stay. Statistics show that 36 billion parcels have been shipped in 2013, 159 billion in 2021, and the expected number of delivered parcels in 2027 is 256 billion [1].

While online shopping is of course a very convenient alternative for customers, it brings challenging logistical problems for the offering companies or their logistics providers. Moreover, particularly in urban areas it comes with a considerably increased traffic related to last-mile delivery services, which collides with CO2 emission objectives in many countries.

For such distribution settings, several approaches to overcome logistical overflow in urban areas are proposed. One of which, that has gained popularity in recent years and is already employed by several companies worldwide (e.g., *Amazon*, *UPS*, *FedEx*), are automated service points or locker box systems, which enable online customers to pick-up their parcels at convenient locations. Also for logistics companies, deliveries to locker systems are rather efficient, as several parcels can be dropped-off at one single location with no risk of a failed delivery due to recipients' absence. However, smart planning of these deliveries is not trivial as companies face a dynamic setting with new orders arriving with expected fulfillment times of only a few days. Furthermore, idle capacities at locker stations are typically scarce as the high installment costs and limited space availabilities force companies to utilize lockers with maximum efficiency.

We address the problem of a company operating parcel locker systems and receiving delivery orders and having to decide whether to accept it (i.e., to deliver the respective parcel to a locker system) or reject it. An example for such a company is the Austrian Post. The assignment of accepted requests can be postponed by a certain number of days, which, however, leads to penalty costs. The acceptance or rejection decision has to be made immediately, while, on a second and time-delayed decision stage, the assignment decision has to be taken. We assume that the latter decisions are taken at the end of each day. The company has to assign the parcel to a specific locker, where customer compatibility has to be respected. A locker is considered compatible if it is located within a maximum radius from the customer. and This part of the decision process has a stochastic component as

☆ Area: Transportation and Logistics. This manuscript was processed by Associate Editor Markus Leitner.

* Corresponding author at: University of Palermo, Department of Engineering, Palermo, Italy.
E-mail address: simona.mancini@unipa.it (S. Mancini).

https://doi.org/10.1016/j.omega.2025.103478
Received 15 April 2025; Accepted 19 November 2025
Available online 25 November 2025
0305-0483/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

idle capacities in locker systems depend on customer pick-up behavior which is not known in advance. The objective of the overall decision problem is to maximize the company's profit, which is determined by revenues for serving requests and penalty payments for delayed service.

To the best of our knowledge, we are the first to introduce the dynamic stochastic decision problem of assigning delivery orders to a system of lockers, where customer pick-up times are stochastic. To generate solutions, we design a decision framework based on a classification approach. A newly-proposed classifier captures the current state of the system and learns from optimal decisions within a deterministic setting. The classifier-based approach is first tested on artificial instances in order to gain insights on solution quality and to benchmark against alternative approaches. The latter include a sequential decision framework based on scenario sampling, which is state of the art for several dynamic stochastic problems. We adapt this approach to our specific problem setting and show that it is outperformed by the classifier-based method. Also classical machine learning (ML) methods such as decision trees as well as different intuitive deterministic policies are examined. All these are outperformed particularly on a real world case, where we use data from the City of Turin, Italy. Several managerial insights are derived in order to support decision making for practitioners.

The rest of our study is structured as follows. We provide a comprehensive literature review in Section 2. The underlying decision problem is mathematically formulated in Section 3, while we provide insights on the sequential decision process in Section 4. The deterministic problem, which is needed to learn from optimal solutions, is given in Section 5 and we describe the complete decision framework in Section 6. The scenario sampling-based approach is presented in Section 6.2. In Section 7 we discuss the computational testbed as well as all numerical results. Our work is summarized in Section 8, where we also highlight future research directions.

2. Literature review

Our literature review is organized as follows. In Section 2.1 we discuss related parcel locker optimization problems. Also, we present different strategies for demand management in the context of out-of-home delivery, such as controlling prices and availability of delivery options. In Section 2.2 we focus on the proposed methodology and review related solution approaches for sequential stochastic optimization problems. Finally, in Section 2.3, we discuss the exploitation of ML approaches to predict solutions for stochastic optimization problems.

2.1. Related parcel locker optimization problems

Out-of-home delivery (OOHD) is a term that indicates all types of unattended deliveries to collection points where the presence of the customer is not required. This mostly refer to delivery to parcel lockers but also include other types of collection point such as shops and stores which gave their availability to receive, and temporary store, parcels for customers. For a survey on OOHD we refer the readers to Janinhoff et al. [2].

Parcel lockers have been identified as a viable solution to reduce operational delivery costs for companies operating in last-mile delivery. The main advantage of deliveries to lockers is the possibility of consolidating orders from different customers, significantly reducing the number of stops required. Furthermore, this delivery system does not require the presence of the customers, and therefore overcomes the risk of unsuccessful and costly deliveries due to for customers' absence. The exploitation of lockers generated new challenging optimization problems, regarding the location and configuration as well as order allocation and the resulting routing decisions. Raviv and Gansterer [3] provide a comprehensive overview of the strategic problems related to the location, capacity, and layout of parcel lockers.

Schwerdfeger and Boysen [4] introduce the concept of mobile parcel lockers, where locker stations can change their locations over the day. Yang et al. [5] model the problem as a bi-level optimization problem, in which the upper-level decides on locker locations, while on the lower-level, customers select the locker which minimizes their pick-up costs.

There are several papers on parcel locker locations, which deal with uncertainty. Most of these works consider uncertainty in customers' demand [6,7].

The first work to consider uncertainty not only on demand but also on capacity availability is Mancini et al. [8]. In this work, the authors consider the fact that since customers have a fixed number of days to collect their parcels, the actual number of days taken by each customer is unknown. This generates uncertainty on the capacity actually available on each locker on a specific day. The problem is modeled as a two-stage stochastic programming model. On the first stage locations decision are made, while the second stage deals with the allocation of customers to lockers. A set of demand and capacity availability scenarios are considered where the goal is to maximize the customers served across all the scenarios.

For what concerns operational decisions, Sitek and Wikarek [9] address a routing problem with alternative delivery options in which customers can be served directly at home or at a compatible delivery point, such as lockers. Models, solution methods, and an analysis on flexible parcel deliveries to lockers are discussed in [10].

Capacity constraints are taken into account but no delivery time windows at customer locations are considered. Mancini and Gansterer [11] introduced a compensation for customers served at a locker and consider time windows for customers served by home delivery. A similar setting has been studied by Grabenschweiger et al. [12], who also considered heterogeneous size boxes and parcels. A multi-period location routing extension of this problem has been proposed in [13]. Dumez et al. [14] study the vehicle routing problem with alternative delivery options with preference levels, in which the customer indicates multiple options, sorted by preference level and the company must respect a minimum level of preference for a given percentage of the customers. Dell'Amico et al. [15] investigate a setting in which lockers may be used not only for delivery but also for picking up returned items.

Galiullina et al. [16] consider the possibility to offer monetary incentives to customers for opting for delivery to lockers. This can be seen as a strategy for demand control and management. Akkerman et al. [17] study a dynamic version of the problem, in which requests arrive during the day and must be processed immediately after their arrival, without having any information about future requests. The company has the possibility to dynamically modify the incentive for opting to OOHD, depending on the current state of the system. The works closer to ours are the following. The first one, Sethuraman et al. [18], considers that pick-up times for parcel are unknown but predict them basing on historical data provided by Amazon. Such predictions are then used to decide the quantity of boxes to reserve for each shipping option (next-day, two-days, standard), as well as the maximum number of orders to accept for each category. The performances of this reservation policy for demand management are then assessed by simulation. Differently from our work, they only distinguish among classes of customers, depending on the selected shipping options, while we deal with the operational decision of acceptance/rejection and allocation of single requests. The second work which is close to ours, is the one presented by Sailer et al. [19], which deals with dynamic acceptance and allocation of parcels considering different compartments and parcels of different sizes. Even if the two problems treated are similar, there are many important differences. First, in their setting only one locker is considered, while we deal with multiple lockers and the selection of the lockers to which allocate the parcel to is a decision variable itself. Second, accepted parcels must be served on the subsequent day, whereas we have a few days to perform the

delivery. This implies that they do not have the postponing option, which make their decision space, simpler than ours. Third, they must guarantee that every accepted parcel can be served on the next day, therefore there is no withdraw and refunding option. Their decision process encompasses two type of decisions: (i) acceptance/rejection which must be taken immediately, and (ii) allocation to a compartment category (small, medium and large), which is performed at the end of the day, where the set of requests accepted over the day is known. Conversely our decision process requires three type of decisions: (i) acceptance/rejection (immediately taken), (ii) fulfilling on next day or postponing, taken at the end of the day, and (iii) allocation to a locker, taken as soon as we decide to fulfill the request on next day.

2.2. Online optimization

Related works in this field mainly concerned approximation algorithms for classical combinatorial optimization problems such as knapsack problems, bin packing, and scheduling problems [20–22].

Powell [23] introduced a unified framework and notation for sequential decision problems. A powerful solution approach consists of taking into account the potential impact of a decision on the future state of the system, by sampling future scenarios, and solving them to extract information that can be used to guide decisions in the current state. A solution framework based on this approach has been introduced and formalized by Van Hentenryck et al. [24] under the name of *online stochastic combinatorial optimization*. This approach has been proven to have very good performances on a broad class of problems. However, it requires considerable computational effort at each decision step, which makes it not suitable for large instances or very complex problems.

Dynamic lookahead policies is a class of methods which, according to the taxonomy provided by Powell [23], exploits scenario sampling and simulation not only to derive a decision at the current state but also to extract information that could guide future decisions for similar states. In most of the related papers simulation is performed in each decision epoch, e.g., [25–27]. Conversely, in [28] all the computational effort is moved to a pre-processing phase where sampled scenarios are analyzed in order to derive a deterministic policy which is then applied at each decision stage. This kind of approach is shown to be very effective and more scalable with respect to other simulation-based approaches.

Other popular classes of approaches for sequential decision problems are *cost function approximation* and *value function approximation*. In the former, an additional term is added to the objective function as an approximation of the expected impact of the current decisions on future states. Many applications of cost function approximations can be found in the literature, such as Deng and Santos [29] which studies an aircraft maintenance scheduling problem and Ghadimi and Powell [30] which addresses an application in energy scheduling. *Value function approximations* estimates the expected reward over all periods for the problem states (e.g. [31]). Finally, *policy function approximations* directly return an action given a state without an additional optimization phase or forecasts on future states [23].

A promising but less researched way is to determine a set of rules to identify potentially risky decisions, that could seem to be good at the moment, but might turn out to be poor in the future as they exclude potentially attractive opportunities. The pioneering work in this field is Romero et al. [32], in which the authors provide a set of rules to quantify the goodness of a decision for *Tetris* in terms of immediate rewards and the potential impact on future rewards. More recently, in [33], a similar idea is applied in the context of online on-demand warehousing, where the authors define a risk indicator which allows to identify requests that, if accepted, could yield a forced rejection of future requests with a consequent negative impact on the total collected profit.

2.3. Machine learning for stochastic optimization problems

The exploitation of ML approaches to solve stochastic optimization problems is constantly growing. Larsen et al. [34] exploit ML to extract information from optimal solutions of two-stage stochastic problems, and use it for predicting second stage costs associated with a given first stage solution, to quickly estimate its impact. Such approach could be very useful in cases where explicitly computing second stage solution is computationally demanding. A similar approach has been applied by Abbasi et al. [35] to predict optimal first stage variable values learning from complete optimal solutions of sample instances. Yilmaz and Büyüktahtakin [36] train predictors on solutions of simple associated deterministic problems, instead of exploiting the solutions of the stochastic problems. Wu et al. [37] use ML to predict the global value of the objective function, where first and second stage costs are considered. ML methods have been also successfully applied to identify more relevant scenarios in order to reduce the number of scenarios needed to be analyzed in a stochastic model and consequently the computational effort required to solve them. For example, Crespo-Vazquez et al. [38] exploit clustering for scenario reduction, while Bengio et al. [39] utilize ML techniques to create a deterministic scenario, more representative respect to the classical equivalent deterministic problem. Differently from the above mentioned methods.

In contrast to the methods mentioned above, our newly proposed approach does not need to analyze full solutions of sequential optimization problems to train ML classifiers. It learns from single decisions even if extrapolated from different solutions. This allows to strongly reduce the number of sample instances to be solved in the training phase and the size of the training set. More precisely, we use the fact that similar situations (system states) may arise in different instances or in the same instance but at different points in time and, even more important, that each decision depends on the current state of the system and not on the set of past decisions that led the system to that state. This allows us to connect the decision only to the current state and to the current exogenous information, neglecting past events and decisions and simplifying both the learning phase and the decision process.

Resuming, our approach exploits the advantage of moving computational efforts of simulation and learning to an a priori phase, like in [28], but at the same time eliminates the need of learning from complete solutions of the stochastic problems, exploiting single decisions to learn how to map states into decisions. Differently from algorithms that need to create a look-up table which maps all possible states into decisions, we create a tool, tuned by ML, which learns to predict the correct decision for a specific state. This approach is particularly efficient when the set of possible states is very large. To the best of our knowledge, we are the first to present such a decision framework and to apply it to a newly proposed problem.

3. Problem description

We consider a set of lockers K, where $k \in K$, each characterized by a capacity C_k that represents the number of locker boxes available. The time horizon T is composed of $|T|$ time slots, each one of which represents a day. A set of customer requests I arrives within the time horizon. Each customer i is characterized by a status, the arrival day a_i (day in which the request entered the system) μ_i, which is either premium or standard, and a geographical location ($\bar{x}_i$ and $\bar{y}_i$).

The revenue achievable serving the request, p_i, and the deadline within which it must be fulfilled, d_i, depend on the status of the request. If a request is accepted it must be delivered to a compatible locker within the deadline, otherwise a penalty δ_i must be paid for each day of lateness. Such lateness, l_i, cannot exceed a maximum value d_i^{max}. If this happens, the customer must be refunded with a compensation r_i. Premium customers yield a higher revenue if accepted, but have a shorter fulfillment time and higher penalties in case of lateness and refunding. Requests dynamically appear and are assumed to be

compatible with lockers located within a given radius. Every time a request is received, the company immediately decides whether to accept it (and get the corresponding revenue) or not.

At the end of a day, the number of boxes currently available in each locker is dynamically updated considering parcels that have been picked up during this day. Afterwards, all requests accepted but not yet fulfilled, are processed. The company decides which request to assign to which locker. Assignment decisions must respect the current available capacity. If some requests cannot be fulfilled due to a lack of space, they are postponed to the next day. Once a parcel has been delivered to a locker, it must be collected within α days. If a customer did not pick-up the parcel within the maximum number of days agreed, the requests is removed from the locker and delivered back to the depot. We assume, without loss of generality, that all customers collect their parcels within the deadline. However, the number of days taken by each customer to collect the parcel is not a priori known to the company. If the number of days elapsed from the arrival of a request exceeds the agreed fulfillment time, the lateness penalty r_i has to be paid. Once the maximum allowed lateness is reached, the customer is refunded and the request is discarded. The goal of the company is to maximize the revenue achieved, which is given by the sum of collected revenues minus the lateness and refunding penalties paid. The values of δ_i, d_i^{max}, and r_i are fixed by the company and uniquely depend on the customer status (premium or standard).

The resulting problem, which we denote as *dynamic stochastic parcel locker assignment with uncertain pick-up times* (DSPLAU), can be modeled as a sequential stochastic optimization problem dealing with three types of decisions: (1) acceptance or rejection of a request, (2) day on which to fulfill the request, and (3) assignment to a specific locker. Note that decisions of type (1) must be taken immediately after the request is received but the company has a given number of days to decide on which day and where the request is assigned to.

4. The sequential decision problem

Leveraging the framework presented in [23], we model the DSPLAU as a sequential decision problem. A decision epoch j is triggered by one of two possible events: (i) Request arrival or (ii) end of the day. The pre-decision state variable S_j encompasses all information necessary to make a decision in j and model the system from j onward.

4.1. Pre-decision state

In case of a request arrival, the state of the system S_j can be exhaustively described by the following elements.

Pending requests. The set of requests accepted but not fulfilled yet, R_j. For each request i, the customer status, the geographical location, and the deadline for fulfillment are known.

Temporal information. This indicates the current day (t_j) and the number of additional premium and standard customer requests, Ω_j and Φ_j, respectively, that are expected to arrive during the day, based on known distributions.

Locker occupancy. For each locker we keep track of the number of boxes that will for sure be idle on the subsequent day, denoted as $\Delta(t_j + g)$, with $g \in \{1, \dots, \alpha - 1\}$. We compute $\Delta(t_j + g)$ as the number of currently empty boxes plus the number of boxes containing parcels delivered exactly $\alpha - g$ days ago as these will be idle on day $t_j + g$. Furthermore, we add the number of expected available boxes due to known distributions of pick-ups within α days. We assume that all lockers are empty in the initial state.

Request. The characteristic of the arrived request i (μ_i, $\bar{x}_i$, $\bar{y}_i$).

In case a decision epoch is triggered by the end of the day, state S_j is described only by two elements: (i) number of pending requests and (ii) locker occupancy, where we compute the number of expected suitable and idle boxes for each non-assigned request in R_j.

4.2. Decision space

The decision space $X(S_j)$ encompasses all feasible decisions in S_j. The decision made in epoch j is identified as χ_j. In case of a request arrival, $X(S_j)$ only includes two alternatives, namely *accept* or *reject*. This decision can be seen as a *demand control* decision. In case of an end of day event, the decision space is much more complex since it includes all possible combinations of assignments of requests to lockers, or their postponement to the subsequent day. Hence, for each request in R_j we have up to $|K| + 1$ feasible decisions (if the request is compatible with all lockers and if all of them have available capacity). Globally, the decision space includes up to $(|K| + 1)^{|I|}$ requests, as each of them can also be postponed. In this case, the decision is denoted as *assignment or postponement* decision. Once a request is assigned, this decision cannot be revised.

4.3. Transition to a post-decision state and exogenous information

After making a decision in S_j, the system deterministically goes to a post-decision state. In case of demand control, the transition consists of a modification of the set of pending requests, i.e., request i, if accepted, is added to R_j. In case of an assignment or postponement decision, the set of pending requests is updated by excluding all the requests (i) that have been allocated to a locker and those (ii) being postponed beyond the fulfillment deadline and therefore are withdrawn. Also, the current locker occupancy for the subsequent α days is consequently updated. The exogenous (stochastic) information addresses the characteristics of the newly arrived request in case of a request arrival event, while encompasses the information about the customers who picked-up their parcel during the current day, in the case of an end-of-day event. The disclosure of the exogenous information marks the transition from a post-decision state to the next pre-decision state.

In each decision epoch triggered by a request arrival, the immediate reward $\bar{R}_j$ depends on the revenue for accepted requests. Hence, $\bar{R}_j = p_i$, where i is the newly arrived request, if the request is accepted, and $\bar{R}_j = 0$ in case of rejection. If the decision epoch is triggered by an end of day event, the associated reward $\bar{R}_j$ is less or equal to 0 and it corresponds to the sum of the lateness penalties occurred on that day plus the penalties for requests withdrawn on that day. Hence, a decision x_j in state S_j generates a contribution $C_j(x_j, S_j)$, where $C_j(x_j, S_j) = \bar{R}_j$.

Our objective is to maximize the expected total contribution starting from an initial state S_0

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{j=0}^{J} C_j(S_j, X_j^{\pi}(S_j))|S_0\right],$$

where decisions are made by using a policy π from the set of decision policies Π. Globally, we want to identify an optimal decision policy $\pi^* \in \Pi$ that maximizes the expected total contribution, given by

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{j=0}^{K} C_j(S_j, X_j^{\pi}(S_j))|S_0\right].$$

5. The oracle model

In this section we provide the mathematical model, named *Oracle*, that describes the decision problem with a complete a priori information on request arrivals and number of days required to collect the parcel for each customer. Note that solutions generated by this model can hardly be used to benchmark the dynamic stochastic decision framework, as for the former we assume the unrealistic case of full information on request arrivals and recipient's behavior over the complete planning horizon. However, we use the model to assess solution quality of the proposed decision framework.

The problem is formulated Integer Programming (IP) model. The value of the optimal solution represents an upper bound on the cumulative reward that can be obtained by the company within the dynamic stochastic decision process. We extend the notation presented above by the following decision variables.

x_i Binary variable taking value 1 if request i is accepted and 0 otherwise
u_i Binary variable taking value 1 if request i is accepted and remained unserved and is therefore withdrawn; 0 otherwise
L_i Lateness of request i (expressed in days, cannot be negative)
y_{ik}^t Binary variable taking value 1 if request i is assigned to locker k on day t; 0 otherwise
z_{ik}^t Binary variable taking value 1 if request i is available in locker k on day t; 0 otherwise

The mathematical model is then formulated as follows.

$$\max \sum_{i\in I} p_i x_i - \sum_{i\in I} r_i u_i - \sum_{i\in I} \delta_i L_i \tag{1}$$

$$\sum_{i\in I} z_{ik}^t \le C_k \quad \forall k \in K, t \in T \tag{2}$$

$$y_{ik}^t \le z_{ik}^{t+h} \quad \forall i \in I, k \in K, t \in T, h \in \{1 \dots \hat{\alpha}_i\} | h + t \le |T| \tag{3}$$

$$\sum_{k\in K} \sum_{t\in T | t \le d_i + d_i^{max}} y_{ik}^t + u_i = x_i \quad \forall i \in I \tag{4}$$

$$L_i \ge \sum_{t\in T} \sum_{k\in K} (t - d_i) y_{ik}^t \quad \forall i \in I \tag{5}$$

$$0 \le L_i \le d_i^{max} \quad \forall i \in I \tag{6}$$

$$y_{ik}^t = 0 \quad \forall i \in I, k \in K, t \in T | t \le t_i^a \tag{7}$$

$$y_{ik}^t = 0 \quad \forall t \in T, i \in I, k \in K | \pi_{ik} = 0 \tag{8}$$

The objective function reported in (1) maximizes the total revenue collected minus the penalties paid for withdrawn requests or lateness of accepted requests. Constraints (2) impose that locker capacities are respected. Constraints (3) imply that if a parcel is delivered to a locker it is stored in that locker until it is collected by the customer, where $\hat{\alpha}_i$ is the number of days taken to pick-up parcel i. Constraints (4) impose that if a request is accepted it is assigned to a locker within the maximum allowed number of days or it is withdrawn. Lateness is computed by constraints (5), while constraints (6) limit the maximum allowed lateness for each request. Constraints (7) and (8) forbid to fulfill a request before it is received (where t_i^a is the arrival time of the request i) or to delivery it to a locker which is not compatible with the customer associated with it, respectively. Note that π_{ik} is an exogenously given compatibility matrix indicating whether request i is compatible with locker x ($\pi_{ik} = 1$) or not ($\pi_{ik} = 0$).

6. Dynamic stochastic decision framework

We assess two different approaches to tackle the dynamic stochastic problem setting. First, we present the newly proposed classification-based method. In order to benchmark it against state-of-the-art we then describe a scenario sampling-based lookahead approach.

6.1. The machine learning-based decision framework

The approach proposed in this work aims to exploit trained classifiers to predict which requests to accept or reject, while a deterministic rule is applied, at the end of each day, to decide which requests to serve and which to postpone. Also the assignment to a locker happens in this stage.

The classifiers are trained on data collected by oracle solutions, i.e., solutions of instances for which we suppose to have perfect information on future requests and pick-up times. These can be obtained by using historical data.

Each single request is considered as a separate record characterized by several features, describing the state of the system at the request arrival and a binary target class, where 1 indicates that the oracle has accepted the request, and 0 that it has rejected it. The training process is illustrated in Fig. 1.

Since the classifier is used only to predict acceptance or rejection of requests and not assignment decisions, only data related to the state of the system on epochs triggered by requests arrivals are considered.

The trained classifier is then applied to instances of the sequential decision problem, for which the information about request arrivals is not available. Each time the platform receives a new request, the trained classifier is applied to take the acceptance or rejection decision. At the end of the day, allocation and postponement decisions are made by a deterministic rule.

The above described decision framework works independently from the underlying classifier. Many classifiers with very good performance are available in the literature. However, the goal of all these classifiers is to maximize the accuracy of the predictions. While this is totally reasonable in a classification problem, it does not necessarily apply once the classifier becomes a tool to solve a sequential decision problem.

In fact, the accuracy of an ML model in replicating oracle decisions, might not be the determining factor in improving stochastic optimization performance. On the contrary, what is crucial is the model's ability to minimize the impact of decision errors on the solution of the stochastic optimization problem. This is particularly true in our case, in which errors made on premium customers have a larger impact on the objective function with respect to those made on standard ones. For this reason, we propose to create a new IP-based classifier which minimizes the weighted sum of errors, where these error are weighted according to the category of the customers.

The proposed classifier relies on the solution of a mixed IP (MIP) model. For this, we assume a set of records I, where each record ($i \in I$) is associated with a request for which the customer status μ_i can be either premium ($\mu_i = 1$) or standard ($\mu_i = 0$). We consider a set of features A, with $a \in A$ such that each record i can be linked to each feature a with a specific value for this feature, b_{ia}.

We assume a set of classes C and for each record i the class to which it belongs, c_i, is known. We consider two classes, namely 1 and 0. Value $c_i = 1$ indicates that the associated request was accepted in the optimal solution of the oracle problem and $c_i = 0$ that it was rejected. We define a parameter β_i, which represents the importance of errors.

Binary variables x_{ic} take value 1 if record i is assigned to class c and 0 otherwise. Weights w_{ac} represent the importance of feature a for class c. These variables are continuous but can assume values only between -1 and 1. A weight equal to 1 means that a very high value on this feature corresponds to an element that most probably belongs to class c, or, in other words, that the feature a and the belonging to class c are strongly positive correlated. Conversely, a weight equal to -1 suggests a strongly negative correlation. A null weight means that this feature does not play an active role on the assignment of this feature to that class. We compute a score value s_{ac} for each combination of feature and class.

Each record must be assigned to exactly one class and it is assigned to the class for which it obtains the highest score. In case of a tie, the assignment is done by the model. The mathematical formulation of the MIP-based classifier is provided in the following.

$$\max \quad \sum_{i\in I} \beta_i x_{ic_i} \tag{9}$$

$$-1 \le w_{ac} \le 1 \quad \forall a \in A, c \in C \tag{10}$$

$$s_{ic} = \sum_{a\in A} b_{ia} w_{ac} \quad \forall i \in I, c \in C \tag{11}$$

$$s_{ic} \ge s_{i\bar{c}} - 2|A|(1 - x_{ic}) + \epsilon \quad \forall i \in I, c \in C, \bar{c} \in C | c \ne \bar{c} \tag{12}$$

$$\sum_{c\in C} x_{ic} = 1 \quad \forall i \in I \tag{13}$$

$$x_{ic} \in \{0; 1\} \quad \forall i \in I \ \ \forall c \in C \tag{14}$$

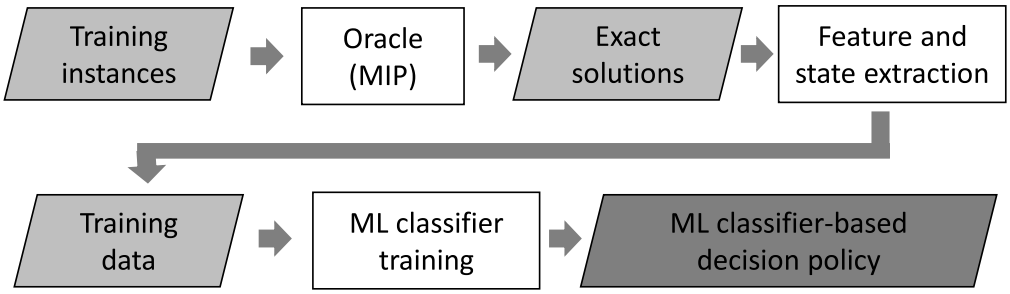

Fig. 1. Illustration of the training process.

The objective function (9) maximizes the number of correctly classified records, i.e., those being assigned to the same class as in the oracle solution. These are weighted by the importance of the associated customer status. We impose by constraints (10) that feature weights can only take values in the range [−1;1]. The score of an item for a specific class s_{ic} is given by the sum of its feature values weighted by feature importance for that class, as expressed in constraints (11). As stated in constraints (12) a record i can be assigned to a class c only if its score related to class c is higher than its score for the second class plus a given threshold ϵ. Larger values of ϵ imply a strong separation among classes but with the risk of obtaining more misclassified records. Each item must be assigned to exactly only one class, as stated in constraints (13). All the features are normalized with a min–max normalization such that they only take values between 0 and 1. Finally, constraints (14) specify the domain of the variables.

The advantage of this classifier is that all the computational effort is relegated to the training part, while the prediction can be obtained within very short computational time. Another important advantage of this method is its complete explainability. In fact, through the values of the w_{jk} variables we can immediately know which are the most important features to be considered in the acceptance decision process and whether their contribution is positive or negative.

For the assignment and postponement decisions, we process all the customers belonging to the set of pending requests, ordered by their fulfillment deadline. Among requests with the same deadline, we process first those associated with premium customers ordered by their arrival time. Each processed request is assigned to a compatible locker with the highest number of idle boxes on the next day. If no available compatible lockers are available, we postpone the request to the next day. Note that unavailability is the only reason for postponing since we are not performing proactive postponements.

6.2. The scenario sampling-based lookahead approach

Scenario sampling is one of the most used techniques in lookahead policies.

In most of the studies, at each epoch, future scenarios of request arrivals are sampled and exploited to derive a decision (e.g., [25,40,41]).

These methods are known to yield very good performance. However, they require a significant computational effort which makes them suitable only for rather simple decision problems or more complex decision problems but with a small number of epochs. Given the very large number of epochs present in realistic sizes of our problem, such approaches are generally not viable. However, we propose to apply the scenarios sampling technique in an a priori phase. To that end, we solve complete request arrival scenarios by considering perfect information. Hence, the oracle can be used to solve each scenario in order to derive lookahead policies to be applied in each epoch of the dynamic stochastic process (see [28]).

Based on this idea, we develop a lookup table tuned by scenarios sampling, where we map each system state with a decision, namely accept or reject. This is based on the most popular decision outcome in situations in which the system was in a specific state. Such systems work well when the number of possible states is limited and the occurrences of each state are enough to be statistically significant. As this is not the case in our problem, we propose to use a simplified version of the system's status, which covers the following three features.

1. Request status (μ_i), which has only 2 categories (premium and standards).
2. Arrival days: early (within two days), medium (between 3 and 8 days), and late (in 9 or more days).
3. Expected available capacity on the next day, compatible with request i which triggered state j, (ξ_j): low (less then 30%), medium (between 30% and 70%), and high (at least 70%) of the total capacity of lockers compatible with a specific request.

Each state corresponds to a combination of values for these 3 features, leading to 18 possible states. By this we can base the lookahead on a small finite number of categorical values, which are used to lookup whether a newly arriving request should be accepted or rejected.

For each epoch, which corresponds to a request arrival, within a scenario, we compute the state of the system and we track the decision made by the oracle model. Then, for this state we determine the most popular decision (accept or reject) in each epoch, where this state occurred. This popular decision is then mapped with the specific state in the lookup table. By this we generate a valid lookup table based on a statistically sufficient number of occurrences for each state. This is then used to derive a decision on each epoch of the sequential decision process. We consider this approach as a benchmark for our newly proposed classification-based decision framework.

7. Computational results

Our computational study consists of two parts. First, we benchmark the results of the proposed MIP-based classification approach (MIP-c) against (i) the oracle solution, (ii) the scenario sampling-based lookahead approach (SSL), (iii) three alternative classification approaches, where classical decision tress with depth 3 (DT3) and depth 5 (DT5) and logistics regression (LR) are applied within the stochastic dynamic sequential decision process, and (iv) the following 4 deterministic policies.

- **Only Premium (OP):** accepts all premium customers and reject all standard ones
- **Premium and fixed number of standard (PFS):** accepts all premium customers and a maximum fixed quantity of standard per day (the first 30% of standard requests received each day)
- **Capacity based (CAP):** accepts a premium customer if the average expected available capacity over the next 3 days is at least 20%, and standard ones if this value is at least 80%
- **Accept all (ALL):** accepts all the requests. This policy replicate what commonly happens in practice, where companies (e.g., Amazon) never reject customers

DT3, DT5, and LR have been implemented in *Orange* and use standard default values for all their hyperparameters.

In the second part of our computational study we focus on key performance indicators computed for the different policies. Also, we exploit insights from applying our solution approach to a real case in the city of Turin in Italy.

7.1. Computational testbed

We generate a set of 10 instances, with 1000 customer requests homogeneously spread over 10 days, where each instance covers one planning horizon. Based on publicly available information,[1] 70% of *Amazon* customers in the United States of America have a premium account. Similar shares can be seen in European countries. Hence, we imply the same distribution. For premium customers we assume a profit of 10, a rejection cost of 15 and a daily delay penalty of 2. The order must be fulfilled within 2 days (without incurring any penalty), while the maximum number of delay days allowed is 3, after which the order is withdrawn and the customer is refunded. For standard customers, profit and rejection costs assume value 2 and 3, respectively. The order must be fulfilled within 5 days, with an allowed maximum delay of 2 days, for which a daily penalty of 1 is paid.

Customers are randomly located in a 100×100 square, the status of the customers is randomly drawn according to a discrete probability function, where the probability of being premium is 70% and that of being standard is 30%. We consider 4 lockers, each containing 35 boxes. Lockers are located symmetrically within the square at points (25,25), (25,75), (75,25) and (75,75). The number of days taken to collect the parcel for a customer is randomly drawn in {1,2,3} with probability 0.4, 0.3, and 0.3, respectively. Each locker contains 35 locker boxes and the compatibility radius between customers and lockers is fixed equal to 50.

The training set is generated by a single instance with the same characteristic as described above. It is composed of 1000 records. Decision trees are generated by *Orange* with a 10-fold cross validation. The MIP-based classifier is implemented in *Xpress 8.13*. Only 100 randomly selected records are used to train the model in order to avoid overfitting.

The value of ϵ is set to 0.005, β_1 to 0.9 and β_2 to 0.1. All the implemented procedures require only a few seconds of running time.

The instances and the codes of the proposed approaches are publicly available at Mancini and Gansterer [42].

7.2. Comparison with benchmark approaches

In Table 1 we benchmark the performance of all tested approaches against the oracle solutions. Note that the presented gaps do not have the same meaning as *optimality gaps* in deterministic optimization problems as they obtained under the unrealistic assumption of full information on requests throughout the whole planning horizon. Besides the gaps we compare the number of premium and standard customers accepted but refunded, i.e., customers that could not be served within the fulfillment deadline.

The results show that the newly proposed approach (MIP-c) strongly outperforms all other approaches including SSL, the two decision tree approaches (DT3 and DT5) and LR, reaching a gap of 8.36% against 19.29%, 16.71%, 16.76%, and 35.14%, respectively. Interestingly, the policy based on accepting all the premium and rejecting all the standard customers (OP) performs better than DT3, DT5, and LR (11.61%) showing that the most important driver for the acceptance decision is the status of the customer. Therefore, it is important to investigate which are the other main drivers to deciding which standard requests to accept. This is the most complex issue in the decision problem.

[1] www.statista.com.

Table 1
Comparison of MIP-c against Oracle, SSL, CAP, DT3, DT5, LR, OP, PFS, CAP, and ALL. We report average percentage gaps of the objective function value and average numbers of accepted and refunded premium (P) and standard (S) customers.

	Gap%	% P accept	% S accept	% P refund	% S refund
Oracle		88.5	34.6	0	0
MIP-c	8.36	86.6	6.4	2.3	0
SSL	19.29	94.2	20.8	14.8	0
DT3	16.76	90.0	20.8	10.5	0
DT5	16.71	89.9	20.8	10.4	0
LR	35.14	59.3	0.3	0	0
OP	11.61	100	0	12.8	0
PFS	29.10	100	33.6	25.4	0
CAP	15.39	100	81.5	12.8	81.5
ALL	16.26	100	100	12.8	100

In fact, simply accepting a fixed percentage of standard customers without differing among them, as done by PFS, yields very poor results (29.10%). This is due to the fact that the high number of standard customers accepted by PFS leads to an overload of the system. While standard customers are more flexible and have later fulfillment deadlines, the system manages to serve them on time, but fails to fulfill a large number of premium customers which can be seen by the high penalties. The behavior of MIP-c is almost the opposite. It is more conservative and tends to accept only customers that can be served with very high probability. This is reflected by a lower number of premium customers (608 over 702 on average) and a very low number of standard customers accepted (only 19 over 298). This generates a lower profit but also the number of customers to refund (all premium) is extremely low, generating a very low amount of penalties. CAP and ALL obtain globally good results, comparable with DT3 and DT5, but much worse than MIP-c and OP. They both accept a very large number of customers, but they are not able to fulfill all of them and this generate very large refunding penalties. While CAP takes into consideration the available capacity and is therefore able to adapt to instances with a different demand/capacity ratio, ALL totally neglects capacity information and obtains therefore good results when this ratio is high, but performs very bad in cases where capacity is a very limited resource with respect to the demand. Therefore, it is not a reliable method. Since ALL represents the strategy currently implemented in practice, where proactive customer rejection is not exploited and all the requests are accepted, this analysis shows the huge benefit achievable by using a smart decision support tool such as MIP-c to deal with this complex decision problem.

SSL, DT3, and DT5 show a similar behavior among each other, i.e., accepting more standard customers with respect to MIP-c but also a higher number of customers to refund, which is index of a low service level. LR, instead, has an opposite behavior, which turns out to be too conservative. In fact, it basically accepts only customers, where the probability that they can be served is very high. This policy, however, yields very low profits. OP accepts all 702 premium customers and rejects all standard ones. However, the number of customers accepted is too high and leads to a high number of refunds (90). PFS empathizes this behavior since in addition to all the premium customers also 100 standard ones are accepted. Again we observe a high number of refunds (178). This strategy leads to poor service levels. Although CAP and ALL obtain good results in terms of global profit, the amount of customers refunded is very high, which reflects a very low quality of service, and reliability of the company, which could yield to a considerable decline in customer demand.

Finally, the optimal solution obtained by Oracle consists of accepting 621 premium customers (a little bit more than MIP-c) and 103 standard ones, without needing to refund any customer. It is worth noting that the conservative behavior of MIP-c might reduce the profit of the company, with respect to the optimal solution, but maintains

Table 2
Feature weights provided by the MIP-based classifier. For each class (accept = 1/reject = 0) we report the weight for the request's status, arrival day (Day), the number of idle boxes for the following 3 days (F1, F2, F3) due to pick-up deadlines and the number of free boxes due to expected pick-ups for the next 3 days (E1, E3, E3). Ω and Φ give the expected arrivals of premium and standard customers, respectively. Diff shows the difference among classes.

Class	Status	Day	F1	F2	F3
1	0.32	−0.49	−1.00	0.38	−0.36
0	−0.53	1.00	−1.00	1.00	−1.00
Diff	0.85	−1.49	0.00	−0.62	0.64
Class	E1	E2	E3	Ω	Φ
1	1.00	1.00	0.18	−0.11	0.24
0	−1.00	1.00	1.00	0.01	−0.01
Diff	2.00	0.00	−0.82	−0.13	0.24

very high service levels. Note that no customers are served late by any approach. This can be explained by the fact that late service is penalized and therefore it is more convenient for the company to refund all customers which orders are already late and use the storage space to fulfill other orders rather than to serve them late with the risk of generating additional lateness penalties on other customers. To avoid this effect, the company could increase the refunding penalty, such that the system prioritizes solutions with more late deliveries but less refunding.

Another important aspect to remark is that all the standard customers are served on time. This is a quite surprising result but can be explained by the fact that the large flexibility of standard customers allows to serve them on time more easier than the premium ones.

Note that it is more convenient to serve a customer i with l days of delay, instead of withdrawing it and to use the same box for a new customer j, only when $p_i - l\delta_i \geq p_i + p_j - r_i$. This obviously depends on specific parameter settings.

It is important to report data on the accuracy, precision, and recall of the 3 best performing classifiers (MIP-c, DT3, DT5). Note that we do not consider LR in this analysis since it performs poorly with respect to the other classifiers. Precision is very high for all the methods (96% for DT3 and DT5 and 95% for MIP-c), but what is more interesting is that DT3 and DT5 have a higher accuracy (94% vs. 84%) as well as a higher recall (94% vs. 86%). Nevertheless, the latter is performing much better in the context of the dynamic stochastic optimization problem. This is an indicator that the most important issue when using classifiers to predict decisions, is not to minimize the number of prediction errors but the impact of these errors on the objective function, as pointed out in [43,44]. To do that, the great advantage of MIP-c is that we can weight the errors made on different types of predictions, while DT3 and DT5 focus only on the accuracy maximization. This makes MIP-c a more powerful and suitable tool to be exploited in a decision framework for sequential stochastic optimization. It is worth noting that we could also use different weighting strategies, e.g., considering larger weights for errors made on accepting customers than those on rejecting. Furthermore, the weighting strategy can be personalized and adapted to different problem structures in different contexts, making MIP-c a very flexible tool that can be easily adapted to several sequential decision problems.

In Table 2 we report the weights of the features for each class provided by the MIP-based classifier. The larger the module of the difference, the higher the importance of that feature in the decision process. In fact, if the module of the difference is 2 the class is extremely important. Conversely when the weight is high for both classes, but the difference is low (near to 0) this means that the feature is not a reliable decision driver.

What emerges from this table is that the main decision driver is the expected number of idle boxes on the next day (E1). This means that if the expected available capacity for the next day is very large, the request should be expected. Although this findings seems intuitive, detailed looks into the results of the classification trees (DT3 and DT5) reveal that this aspect is overlooked by them. There, the main driving factor is the status of the customer. In fact, the status is certainly relevant and correlated with the acceptance decision. Surely, the probability that the optimal choice is to accept the customer should be high if the customer is premium and lower if the customer is standard. However, there is a high number of premium customers that should be rejected (about 11%) and of standard ones that should be accepted (about 33%).

We further observe that the arrival day negatively correlates with an acceptance decision meaning that the larger the day, the lower the probability that acceptance is a good choice. Interestingly. the information about the capacity which will be certainly available (F1, F2, and F3) on a certain day is only very marginally relevant. In fact, the capacity which we expect to be available (E1, E2, and E3) has clearly stronger impact.

We believe these insights are very useful both for the academic community and for the practitioners and that one great advantage of MIP-c is the capability of providing such detailed information about the features' relevance.

Most importantly, it should be emphasized that the strength of this study is not to analyze the real weights of the features as they clearly might differ with different prices or business contexts. The main contribution is to show that the newly proposed MIP-based classifiers outperforms all alternative approaches used within the dynamic stochastic optimization problem.

Table 3
Tuning of β_i errors' weights.

β_i *premium*	β_i *standard*	Gap
1.0	0	17.00%
0.9	0.1	8.36%
0.8	0.2	21.47%
0.7	0.3	24.25%
0.6	0.4	43.71%

7.3. Importance of error weights selection for the classifiers

In the MIP-c classifier, we have the possibility to give different weights β_i to different errors. This is to guide the classification method to minimize the impact if the error on the objective function value rather than the number of errors made (as in standard classifiers). The choice of these weights plays a crucial role in the performance of the algorithm. Since we know that premium customers have larger revenues and penalties than standard ones, and that, for this reason, a misclassification has a potentially larger impact on the objective function, we exploit this knowledge assigning larger weights to premium requests than to standard ones. Here we conduct a sensitivity analysis in which we report the average gap from the oracle obtained with different weight combinations. Results are reported in Table 3 from which it can be seen that the best combination is to consider $\beta_i = 0.9$ for premium customers and $\beta_i = 0.1$ for standard ones.

As can be noticed from the table, errors weights play a crucial importance for MIP-c. For this reason, we conduct another experiment, in which we apply the same weights exploited by MIP-c to a weighted decision tree to see if the weights choice has a beneficial impact also for DT3 and DT5. This, however, is not the case as the performance only slightly affects DT5 (from 16.76% to 16.71% from the Oracle) and remains the same (16.71%) for DT3. This result shows that decision tree performances cannot be significantly improved by considering the potential effect of the errors on the objective function, while MIP-c greatly benefits from this information. We further exploit an automatic error weighting tool based on the SPO+ framework (*smart-predict-then-optimize*) provided by Elmachtoub and Grigas [45], which starts

from homogeneous weights and at each iteration computes the loss generated by these weights such as the difference between the obtained objective function value and the upper bound provided by the oracle solution. It then modifies the weights accordingly by means of the gradient method. The procedure terminates once the loss overcomes a minimum threshold (which we fix equal to 1% from the Oracle) or after a fixed number of iterations (100 in our case). More in details, SPO+ solves at each iteration the model in (9)–(14), with a given combination of β_i variables, provided by the application of the gradient method. The training set is exactly the same used for MIP-c in order to ensure a fair comparison. The only difference is that, while in MIP-c weights are fixed according to the priority status of the customer (higher weights for premium customers), in SPO+ they are iteratively optimized by means of the gradient method.

Although SPO+ has been shown to be very effective in the literature and is currently considered the state-of-the art for integrated prediction and optimization tasks, it performs quite poorly for our problem (18.40% from the Oracle against 8.36% achieved by MIP-c). Our explanation for this performance is that SPO+ is a very effective tool to tune error weights when no error impact information is available. However, in cases like ours, where error impact can be approximated (e.g., premium customers have a more relevant impact than standard ones), exploiting this knowledge allows to strongly improve the results from an optimization point of view. In fact, MIP-c exploit in a very effective way the information about customers status which is, instead, ignored by SPO+.

7.4. Importance of features

The scope of this analysis is to evaluate whether the features with larger weights play an essential role for ensuring a good performance of the MIP-c classifier. For this, we conduct three different experiments. In the first one, we remove the feature associated with the customer status, in the second one, we remove the feature related to the arrival day, whereas in the third one we remove the feature associated with the expected capacity on the next day (E1). This selection is based on the observation that these features are the most influential ones (see Table 2). The first two experiments lead to similar results. By removing the status, the gap grows up to 28.95% while removing the second feature it grows up to 29.38%. This shows that both features are not only important, but also essential for guaranteeing a good performance of the algorithm. Conversely, removing E1, we obtain only a slight decrease in performances, with a gap of 10.17% from the Oracle. Hence, also E1 has a clear impact but is less essential than the other two. This insight helps to improve the algorithm's performance.

7.5. Impact of delivery deadline on profits and customer satisfaction

In this analysis we quantify the impact on the delivery deadline for premium and standard customers on the company's profit and on customer satisfaction (expressed in terms of percentage of orders accepted). We first modify the deadline for premium customers, while keeping the one for standard ones constant. Three different values are analyzed, namely 1, 2, and 3 days. Profit achievable and percentage of total, premium, and standard customers accepted, are reported in Table 4. These tests are carried out by the Oracle model, in order to be able to compare optimal solutions. What emerges from this analysis is that ensuring delivery on the next day for premium customers, which would be perceived as a very high quality of service, decreases the profit by 2.54% and the number of customers served by 3.3%. This impact is stronger for standard customers (−6.7% served) than on premium ones (−1.8%). Conversely, if we increase the delivery deadline for premium customers to 3 days, and by this decreasing the service quality, the profit increment achieved is very low (0.32%). The number of premium customers served remains constant while an increment of 3.5% is observed on the number of standard customers accepted. Globally, we see that advantages in increasing the deadline are marginal for the company. At the same time, the profit reduction incurred by ensuring next-day delivery is only partially justified by the small percentage increment of customers satisfied. Therefore, we state that the most appropriate deadline is the one assumed in the other parts of our computation study (2 days), which guarantees a good balance between customer satisfaction and company profit. We conduct the same analysis varying the deadline for standard customers (in combination with all the premium deadlines) but this parameter does not have any impact on the solution. Hence, we can state that reducing or increasing standard customer deadlines, does not affect neither the number of customers served nor the profit. However, ensuring a service within 4 days, instead of 5, would increase the service quality perceived by standard customers. This can be realized for almost no costs and is therefore a very attractive option. However, one can argue that reducing the difference of treatment (in terms of deadlines) between premium and standard customers, could incentive them to not enter the premium status. This would obviously lead to lost profits for the company. Nevertheless, note that the most relevant benefit of holding a premium status is the percentage of acceptance (88.5% against 37.9%), which can be a convincing incentive for customers to maintain their premium status.

Table 4
Profit and percentage of total premium and standard customers accepted for different premium customers' deadlines.

Premium deadline	Profit	% accepted	% P accepted	% S accepted
1	6251.4	69.1%	86.7%	27.7%
2	6414.4	72.4%	88.5%	34.4%
3	6435.2	73.4%	88.5%	37.9%

7.6. Application to real case

To test our approach in a real world setting, we apply it to data from the city of Turin, Italy.

In our analysis, we focus only on the southern part of the city and we use *Google maps* to derive the addresses of 40 already installed locker stations. For each district, the city of Turin provides information about the number of inhabitants of a certain age class,[2] where 4 age classes are considered (0–17, 18–30, 31–45, 46–65, and >65).

The age distribution of the population is of high relevance for our study as the willingness to accept deliveries, and thus the demand in our case, clearly varies among age classes.

In our work, we use data collected by a survey conducted in Austria, which states that the percentage of customers willing to pick-up deliveries from lockers is significantly higher in age classes below 65 years [8], namely 67%, 71%, 68% and 71%. This percentage strongly decreases for people aged >65 (47%). Combining data about population distribution and willingness to use lockers, we derive the probability of a request to come from a specific district. This data is then used to generate realistic instances, each of which comprises 1000 requests spread over a time horizon of 10 days. The probability that a customer is premium is set to 70% (basing on data provided by www.statista.com) and the pick-up time is randomly selected among 1, 2, and 3 days, with probabilities 0.4, 0.3. and 0.3, respectively. Distances among locations are computed using Manhattan distances. Since Turin is characterized by a set of orthogonal streets, which makes its map similar to the Manhattan one, using this distance metric seems reasonable. A sample instance is depicted in Fig. 2.

In Tables 5 and 6 we report a comparison of MIP-c and DT3 used within the sequential decision framework as well as optimal results obtained by the oracle. Also, the results reached by SSL are provided.

[2] http://aperto.comune.torino.it/.

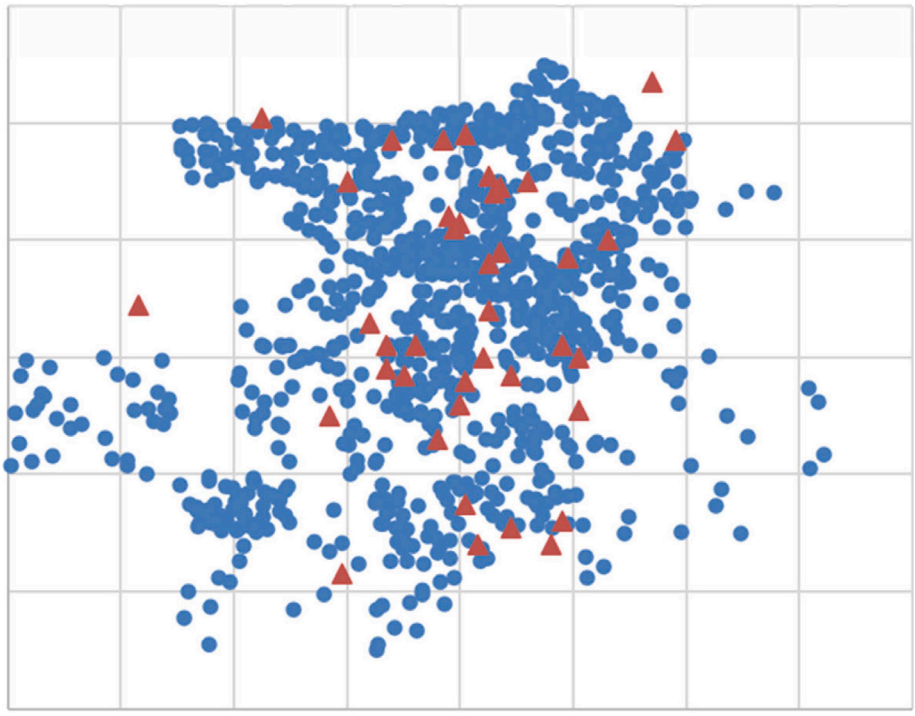

Fig. 2. Instance layout for the real case. Customers are depicted as blue circles, lockers as red triangles.

Table 5
Real case results with small compatibility radius (0.5 km).

	Gap	P accept	S accept	P refund	S refund
Oracle	0.00%	84.11%	83.36%	0.00%	0.00%
MIP-c	2.05%	87.17%	64.33%	2.51%	0.00%
SSL	5.07%	96.72%	20.77%	7.95%	12.74%
DT3	4.07%	94.57%	94.67%	9.80%	10.62%

Table 6
Real case results with large compatibility radius (1 km).

	Gap	P accept	S accept	P refund	S refund
Oracle	0.00%	90.11%	89.20%	0.00%	0.00%
MIP-c	5.53%	100.00%	99.90%	9.89%	10.71%
SLL	5.33%	98.01%	20.77%	4.79%	7.05%
DT3	5.53%	100.00%	100.00%	9.89%	10.80%

We examine tow different compatibility radii, namely 0.5 km and 1 km. Again we collect data about the percentage of premium and standard customers accepted and refunded.

The analysis on the small compatibility radius (Table 5) provides very interesting findings. MIP-c is able to reach near optimal solutions (only 2.05% from the oracle) accepting around 80% of the customers. Only a very small percentage of premium customers are refunded (2.5%) while all standard customers requests are fulfilled. Both SLL and DT3 show lower solution quality. In case of DT3 only 94% of the customers are accepted among which 10% must be refunded due to lack of capacity. SLL accepts only 20% of standard customers and about 7% have to be refunded.

The lesson learned from this analysis is that even with small compatibility radii very high acceptance rates and customer satisfaction levels can be reached by the proposed solution approach (MIP-c).

Results with a large compatibility radius (Table 6) are less surprising as the oracle already shows that almost all customers (about 90%) can be accepted and served on time. Both MIP-c and DT3 suggest to accept all the customers, which, however, leads to 10% of them having to be refunded.

8. Conclusions and future developments

In this paper we introduced a new and relevant decision problem arising in last-mile delivery, named Dynamic Stochastic Parcel Locker assignment with uncertain pick-up times (DSPLAU). The problem consists of two decision phases. First, acceptance or rejection decisions of dynamically arriving customer orders have to be taken, while in a second stage, accepted orders have to be assigned to a set of geographically dispersed lockers. The company has a fixed number of days within which the deliveries have to be performed. A penalty has to be paid for each day of delay. A maximum lateness is allowed after which the request is withdrawn and the customer has to be refunded. The number of days taken by a customer to pick-up the parcel is not known a priori. Clearly, the customers' behavior determines uncertainty on the capacity available for the next days. The problem is modeled as a sequential stochastic decision process, in which three type of decisions are considered.

To solve the problem, we proposed an innovative classification-based approach, which learns from the decisions made by an exact model. The classifier builds on the state of the system and the characteristics of the newly arrived requests and is able to predict whether it is better to accept or reject it. The novelty of this MIP-based classifier with respect to standard ones from the literature is that it is able to weight errors made on different type of requests in regards of the impact that the error can have on the objective value. Furthermore, it provides a total explainability of the obtained results, allowing to determine the features that have a strong impact during the acceptance or rejection decisions.

In our computational study, we used the proposed MIP-based classifier within a dynamic stochastic solution framework and compared it against several alternative approaches. These included a lookahead method, which is based on scenario sampling and standard decision trees with different depths which we used within our dynamic stochastic decision framework. Also, four different deterministic policies were applied. Benchmarking was done both on artificial instances and on a real world case.

The results showed that our approach strongly outperforms all other methods particularly on instances, where customers are accepting rather small compatibility radii for picking-up parcels, which we consider the practically more realistic case.

The key of success of our approach is the possibility of weighting prediction errors not such that the number of errors made is minimized (as standard classifiers do) but that the impact of these errors on the solution of the online optimization problem is minimized. In particular, we assign much larger weights to errors on premium customers since their impact on the objective function is higher. We compared our approach with weighted decision trees, to which we passed the same weights we used and showed that, while the usage of weights allows to enhance the performance of our classifiers, they give only a very low benefit when embedded in a decision tree. Furthermore, we compared with the well-known SPO+ framework. Results showed that we strongly outperform SPO+. This is probably due to the fact that we exploit knowledge on the impact of different errors (premium customers impact more than standard), while SPO+ tries to estimate the impact with an iterative procedure neglecting the obvious information.

Moreover, we exploited the obtained results to derive insights. The main finding were that the feature which plays the most important role in the acceptance decision is not the status of the customer (premium or standard) as stated by the decision tree, but the expected available capacity on the next day. From the real world case we learned that using the proposed solution approach enables the acceptance of a very large percentage of customers, with only a very small percentage of refunded customers. This holds even if only small compatibility radii for the lockers are assumed. By this, we yield a high perceived service quality which increases customer satisfaction and the success of the platform. Another important insight is that ensuring next-day delivery (instead of delivery within 2 days) for premium customers yields to a loss of profit by 2.54% and a considerable reduction (more than 3%) of customers served. This percentage is doubled when looking at standard customers. Ensuring service within 3 days instead of 2, which is a reduction of the perceived quality of service, returns a very low increment of profit (only 0.32%). For what concerns standard customers, reducing their delivery deadline from 5 to 4 days yields an increment of service quality without any profit loss.

From an application point of view, future developments could concern the consideration of different sizes of boxes and parcels. It would also be worthwhile to address a robust version of the problem in which, to accept a request, fulfillment within given deadlines needs to be guaranteed. From a methodological point of view, the newly proposed decision framework might generalized and adapted to effectively solve a broad class of other dynamic stochastic sequential decision problems.

CRediT authorship contribution statement

Simona Mancini: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Margaretha Gansterer:** Writing – original draft, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Statista. Global parcel shipping volume between 2013 and 2027 (in billion parcels). 2022, https://www.statista.com/statistics/1139910/parcel-shipping-volume-worldwide/.

[2] Janinhoff L, Klein R, Sailer D, Schoppa J. Out-of-home delivery in last-mile logistics: A review. Comput Oper Res 2024;168:106686.

[3] Raviv T, Gansterer M. Location, capacity, and layout problems of automated parcel lockers. In: Reference module in social sciences. Elsevier; 2025.

[4] Schwerdfeger S, Boysen N. Optimizing the changing locations of mobile parcel lockers in last-mile distribution. European J Oper Res 2020;285(3):1077–94.

[5] Yang G, Huang Y, Fu Y, Huang B, Sheng S, Mao L, Huang S, Xu Y, Le J, Ouyang Y, et al. Parcel locker location based on a bilevel programming model. Math Probl Eng 2020;1:5263689.

[6] Bieniek M. A note on the facility location problem with stochastic demands. Omega 2015;55:53–60.

[7] Pagès-Bernaus A, Ramalhinho H, Juan AA, Calvet L. Designing e-commerce supply chains: a stochastic facility–location approach. Int Trans Oper Res 2019;26(2):507–28.

[8] Mancini S, Gansterer M, Triki C. Locker box location planning under uncertainty in demand and capacity availability. Omega 2023;120:102910.

[9] Sitek P, Wikarek J. Capacitated vehicle routing problem with pick-up and alternative delivery (CVRPPAD): model and implementation using hybrid approach. Ann Oper Res 2019;273:257–77.

[10] Orenstein I, Raviv T, Sadan E. Flexible parcel delivery to automated parcel lockers: models, solution methods and analysis. EURO J Transp Logist 2019;8(5):683–711.

[11] Mancini S, Gansterer M. Vehicle routing with private and shared delivery locations. Comput Oper Res 2021;133:105361.

[12] Grabenschweiger J, Doerner K, Hartl R, Savelsbergh M. The vehicle routing problem with heterogeneous locker boxes. Central Eur J Oper Res 2021;29:113–42.

[13] Grabenschweiger J, Doerner K, Hartl R, Savelsbergh M. The multi-period location routing problem with locker boxes. Logist Res 2022;15:1–25.

[14] Dumez D, Lehuede F, Peton O. A large neighborhood search approach to the vehicle routing problem with delivery options. Transp Res Part B: Methodol 2021;144:103–32.

[15] Dell'Amico M, Montemanni R, Novellani S. Pickup and delivery with lockers. Transp Res Part C: Emerg Technol 2023;148:104022.

[16] Galiullina A, Mutlu N, Kinable J, Van Woensel T. Demand steering in a last-mile delivery problem with home and pickup point delivery options. Transp Sci 2024;58(2):454–73.

[17] Akkerman F, Dieter P, Mes M. Learning dynamic selection and pricing of out-of-home deliveries. Transp Sci 2025;59(2):207–450.

[18] Sethuraman S, Bansal A, Mardal S, Resende M, Jacobs T. Amazon locker capacity management. INFORMS J Appl Anal 2024;54(6):455–70.

[19] Sailer D, Klein R, Steinhardt C. Dynamic demand management for parcel lockers. 2024, arXiv preprint arXiv:2409.05061.

[20] Coffman E, Csirik J, Galambos G, Martello S, Vigo D. Handbook of combinatorial optimization. In: Du D-Z, Pardalos PM, Graham RL, editors. Chapter bin packing approximation algorithms: Survey and classification. 2013, p. 455–531.

[21] Christensen H, Khan A, Pokutta S, Tetali P. Approximation and online algorithms for multidimensional bin packing: A survey. Comput Sci Rev 2017;24:63–79.

[22] Diedrich F, Jansen K, Schwarz U, Trystram D. A survey on approximation algorithms for scheduling with machine unavailability. Lecture Notes in Comput Sci 2009;5515:50–64.

[23] Powell WB. Reinforcement learning and stochastic optimization. John Wiley & Sons, Ltd; 2022.

[24] Van Hentenryck P, Bent R, Upfal E. Online stochastic optimization under time constraints. Ann Oper Res 2010;177(1):151–83.

[25] Brinkmann J, Ulmer M, Mattfeld D. Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. Comput Oper Res 2019;106:260–79.

[26] Thul L, Powell W. Stochastic optimization for vaccine and testing kit allocation for the COVID-19 pandemic. European J Oper Res 2021;304:325–38.

[27] Ausseil R, Ulmer MW, Pazour J. Online acceptance probability approximation in peer-to-peer transportation. Omega 2024;123:102993.

[28] Mancini S, Ulmer M, Gansterer M. Dynamic assignment of delivery order bundles to in-store customers. Omega 2025;133:103246.

[29] Deng Q, Santos B. Lookahead approximate dynamic programming for stochastic aircraft maintenance check scheduling optimization. European J Oper Res 2022;299:814–33.

[30] Ghadimi S, Powell W. Stochastic search for a parametric cost function approximation: Energy storage with rolling forecasts. European J Oper Res 2024;312:641–52.

[31] Ulmer MW, Soeffker N, Mattfeld DC. Value function approximation for dynamic multi-period vehicle routing. European J Oper Res 2018;269(3):883–99.

[32] Romero VIM, Tomes LL, Yusiong JPT. Tetris agent optimization using harmony search algorithm. Int J Comput Sci Issues 2011;8:1–22.

[33] Ceschia S, Gansterer M, Mancini S, Meneghetti A. Solving the online on-demand warehousing problem. Comput Oper Res 2024;170:106760.
[34] Larsen E, Lachapelle S, Bengio Y, Frejinger E, Lacoste-Julien S, Lodi A. Predicting tactical solutions to operational planning problems under imperfect information. INFORMS J Comput 2022;34(1):227–42.
[35] Abbasi B, Babaei T, Hosseinifard Z, Smith-Miles K, Dehghani M. Predicting solutions of large-scale optimization problems via machine learning:A case study in blood supply chain management. Comput Oper Res 2020;119:104941.
[36] Yilmaz D, Büyüktahtakin I. A non-anticipative learning-optimization framework for solvingmulti-stage stochastic programs. Ann Oper Res 2024. http://dx.doi.org/10.1007/s10479-024-06100-7, forthcoming.
[37] Wu Y, Song W, Cao Z, Zhang J. Learning scenario representation for solving two-stage stochastic integer programs. In: International conference on learning representations. 2021.
[38] Crespo-Vazquez JL, Carrillo C, Diaz-Dorado E, Martinez-Lorenzo JA, Noor-E-Alam M. A machine learning based stochastic optimization framework for a wind and storage power plant participating in energy pool market. Appl Energy 2018;232:341–57.
[39] Bengio Y, Frejinger E, Lodi A, Patel R, Sankaranarayanan S. A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In: International conference on integration of constraint programming, artificial intelligence, and operations research. 2020, p. 99–111.
[40] Van Hentenryck P, Bent R, Mercier L, Vergados Y. Online stochastic reservation systems. Ann Oper Res 2009;171(1):101–26.
[41] Hentenryck V, Pascal, Bent, Russell, Upfal, Eli. Online stochastic optimization under time constraints. Ann OR 2010;177:151–83.
[42] Mancini S, Gansterer M. Instances and codes for dynamic stochastic parcel locker assignment with uncertain pick-up times. Mendeley Data 2025;V1. http://dx.doi.org/10.17632/x66jfgthxd.1.
[43] Mandi J, Kotary J, Berden S, Mulamba M, Bucarey V, Guns T, Fioretto F. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. J Artif Intell Res 2024;81:1623–701.
[44] Sadana U, Chenreddy A, Delage E, Forel A, Frejinger E, Vidal T. A survey of contextual optimization methods for decision-making under uncertainty. European J Oper Res 2025;320(2):271–89.
[45] Elmachtoub A, Grigas P. Smart "predict, then optimize". Manag Sci 2022;68(1):9–26.