A Markov decision process approach for managing medical drone deliveries

Amin Asadi\textsuperscript{a,b,*}, Sarah Nurre Pinkley\textsuperscript{b}, Martijn Mes\textsuperscript{a}

\textsuperscript{a} University of Twente, Department of Industrial Engineering and Business Information Systems Enschede, 7522 NB, The Netherlands
\textsuperscript{b} University of Arkansas, Department of Industrial Engineering, 4207 Bell Engineering, Fayetteville, AR 72701, United States of America

\textbf{A R T I C L E I N F O}

\textbf{Keywords:}
Markov decision processes
Drones
Healthcare
Routing
Dynamic scheduling allocation
Reinforcement learning

\textbf{A B S T R A C T}

Drone delivery is a fast and innovative method for delivering parcels, food, and medical supplies. Furthermore, this low-contact delivery mode contributes to reducing the spread of pandemic and vaccine-preventable diseases. Focusing on the delivery of medical supplies, this paper studies optimizing the distribution operations at a drone hub that dispatches drones to hospitals located at different geographic locations. Each hospital generates stochastic demands for medical supplies to be covered. This paper classifies stochastic demands based on the distance between hospitals and the drone hub. Satisfying the demands requires flying over different ranges, which is directly related to the amount of charge of the drone batteries. We develop a stochastic scheduling and allocation problem with multiple classes of demand and model the problem using a finite Markov decision process approach. We provide exact solutions for the modest sizes instances using backward induction and discuss that the problem suffers from the curses of dimensionality. Hence, we provide a reinforcement learning method capable of giving near-optimal solutions. We perform a set of computational tests using realistic data representing a prominent drone delivery company. Finally, we analyze the results to provide insights for managing drone hub operations and show that the reinforcement learning method has high performance compared with the exact and heuristic solution methods.

1. Introduction

During the last decade, there has been substantial growth in the use of drones for various applications, including but not limited to transportation, agriculture, and delivery (DHL, 2016; Jensen, 2019; Mutzabaugh, 2017; Weise, 2017). Specifically, delivery using drones has received extensive attention as it can reduce air pollution and traffic in congested areas (Ohote & Limbourg, 2020). Moreover, drones are a viable option to reach remote locations with inadequate road infrastructure (Davit, 2019). During pandemics, drones provide a safe and low contact delivery method, which can effectively slow down the spread of the diseases (McNabb, 2020; UNICEF Supply Division, 2020).

Many companies and organizations, including Vanuatu’s Ministry of Health and Civil Aviation (Kent, 2019), Zipline (Lyons, 2020), Matternet (Matternet, 2020), and Manna Aero (Chandler, 2020) use drones to deliver and distribute medical supplies such as vaccines, medicine, and blood units. Effective operations of a fleet of drones require incorporating battery constraints, including limited flight range, time-consuming charging operations, and the high price and short lifetime of batteries. In this research, we provide a model that effectively uses the charge of the batteries to maximize the amount of stochastic demand met. Specifically, we classify the stochastic demands according to drones’ flight range, which depends on the charge inside drone batteries. We ultimately maximize the expected total met demand for delivering medical items using drones dispatched from a battery swap station located in a drone hub.

Battery swap stations are a solution to alleviate the aforementioned issues. In battery swap stations, charged batteries are swapped with empty batteries in short minutes. For instance, Matternet provides a station to automatically swap drone batteries used for delivering blood units and medicine between the supplier and hospitals (Ball, 2020). Besides the quick swapping operation, recharging batteries in anticipation of demand can reduce overcharging and fast charging of batteries, which are shown to accelerate the battery degradation process (Lacey et al., 2013; Shirk & Wishart, 2015). The faster batteries degrade, the quicker the need for battery replacement, leading to a higher cost and environmental waste. The application of a battery swap station is not limited to drones and can be extended to electric vehicles (Fusheng, 2019; Lambert, 2018), electric scooters (BBC, 2015) and cell phone battery swaps in airports, hotels, and amusement parks (FuelRod, 2017). Notably, the number of electric vehicle swap stations is growing in different regions of the world (Chauvet, 2015; Dongmei, 2016; Gordon-Bloomfield, 2014; PeerMohamed, 2017).

In this research, we consider a swap station located at a drone hub that dispatches drones to satisfy...
multiple classes of stochastic demand for medical supplies, which are classified based on their distance from the station.

Given the growth in the number and applications of battery swap stations, it is crucial to optimally manage the stations’ operations to reach the highest performance of the station. Thus, we design a decision-making framework to provide optimal recharging and distribution policies when considering the stochastic demand originating from different geographical locations. The drone hub can send drones to locations within their flight range, which differ based on the level of charge inside their batteries. It is a complicated setting, given that the level of charge inside batteries can be any number between 0 and 1, and different combinations of charge levels can be used to satisfy the stochastic demands generated from places located at different distances from the drone hub. Hence, using aggregation and discretization, we classify the demand based on the distance between the hub and demand locations. To the best of our knowledge, we are the first to use such classification for this problem and link a level of charge of batteries with a class of demand such that the demand of each class can be satisfied with batteries having the same or higher level of charge. That is, each class of demand can be satisfied with one or multiple levels of charge of batteries. We formulate this problem as a stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD).

We model the stochastic SA-MCD as a Markov decision process (MDP). It is an appropriate modeling approach for problems like ours that are in the class of sequential decision-making under uncertainty (Puterman, 2005). The decisions are made in discrete points in time or decision epochs. We represent the state of the system in the MDP as the number of batteries within each charge level class. The actions of the MDP are the number of batteries recharging from one level to an upper level of battery charge. The transition probability is a complex function governed by multiple classes of stochastic demand. In our MDP, the optimal policy determines the maximum expected total reward, which is a function of total weighted met demand of different classes.

We use backward induction (BI) and a reinforcement learning method with a descending ε-greedy exploration feature (DrRL) to solve the SA-MCD. The exploration feature allows the algorithm to take random actions to assist in escaping local optima by visiting apparently non-attractive states. The descending exploration means that as the algorithm proceeds in iterations, the probability of taking an arbitrary action decreases, and the algorithm performs more exploitation (taking favorable actions) by visiting attractive states. BI can provide exact solutions for problems like the SA-MCD that have finite state and action spaces (Puterman, 2005). However, BI runs into the curses of dimensionality and faces computational time and memory issues as our problem size increases. Thus, we apply an RL method that is able to find high-quality approximate solutions for large-scale SA-MCDs, which are not solvable using BI (Powell, 2011). We show the convergence of our RL method and its low computational time and memory requirement.

We computationally test the SA-MCD model and solution methods on a case study related to the Zipline drone delivery company, which delivers blood units, vaccines, and medical supplies in Rwanda. We consider the drone delivery of these supplies from its station located in the Muhanga district, Rwanda, to the reachable hospitals throughout the country. We consider the population of districts, number of hospitals in each district, number of people using a hospital, and rate of arrivals to each hospital to find the stochastic orders for medical supplies. Then, we convert the orders to the demand for drone missions, given that each drone can carry 2 kg of medical products at a time (Baker, 2017). For our computational experiments, we classify this demand and import the real data associated with the distance between locations, the population of districts, flight regulations in Rwanda, and the Zipline drone configuration, including the speed, flight range, and recharging time.

We derive insights from solving the SA-MCD to manage the distribution operations of the swap station using different sets of computational experiments. We provide the optimality gap and average percentage of the met demand using the DrRL method for modest problem sizes (15–21 drones). Results show that the Zipline company needs more drones to satisfy 100% of the stochastic demand. Hence, we draw the relationship between the number of drones in the station and the amount of met demand using our RL solution for larger instances of SA-MCD. We also analyze the interplay between the different demand classes and the use of higher-level charged batteries to satisfy lower-class demand. Finally, we more generally validate the performance of our RL method by considering a fictitious case.

**Main Contributions.** We summarize the main contribution of this paper as follows.

- We propose a stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD) for managing operations of a drone swap station located at a drone hub. We classify the demand based on the distance between the station and hospitals generating the stochastic demands.
- We develop an MDP model for the SA-MCD and solve small instances of SA-MCD using backward induction (BI). We show the inability of BI to solve large-scale SA-MCDs required for the real application.
- We propose a descending ε-greedy reinforcement learning method (DrRL) to find optimal and near-optimal policies for the station that faces stochastic demand for sending drones to deliver medical supplies. We show high performance of DrRL for solving the SA-MCD.
- We leverage our model and our approximate solution method to be used for a real application, which is the case study related to Zipline medical supply delivery using drones in Rwanda.
- We conduct different sets of experiments to derive insights for managing the operations in a swap station to maximize demand satisfaction. We show that demand classification approach in modeling improves demand satisfaction, which is the primary purpose of delivering medical supplies using drones. Furthermore, to provide generic insights, we analyze the performance on both the Zipline case as well as a fictitious case.

The remainder of this paper is organized as follows. In Section 2, we discuss relevant literature related to the modeling, application, and solution methods for the stochastic SA-MCD. In Section 3, we present our Markov Decision Process to model the stochastic SA-MCD. In Section 4, we discuss the exact and approximate solution methods. In Section 5, we outline the computational experiments conducted and provide insights for managing swap station operations. We end with concluding remarks and propose directions for future research in Section 6.

2. Related work

There is a growing interest in the use of drones for various applications. We provide an overview of scientific works, which are relevant to the model, application, and solution methods presented in this paper. Therefore, we focus on providing an overview of research related to managing operations in swap stations, delivering medical items using drones, Markov Decision Process (MDP) modeling for dynamic problems, demand classification, and reinforcement learning (RL) methods.

Various researchers have studied managing swap station operations. Asadi and Nurre Pinkley (2021) present an MDP model to find the optimal/near-optimal policies (number of recharging/discharging and replacement actions) to maximize the expected total profit for the station facing stochastic demands and battery degradation. They solve this problem using a heuristic, RL methods, and a monotone approximate dynamic programming algorithm (Asadi & Nurre Pinkley,
2022) to provide insights for managing the internal operations in the swap stations. Widrick et al. (2018) propose an MDP model for the same problem when no battery degradation is considered. Nurre et al. (2014) do not consider stochasticity and provide a deterministic model to find the optimal policies for managing swap stations. Our work is fundamentally different from the discussed papers as they do not consider different demand classes and multiple states of charge of batteries. Besides, our objective is satisfying the amount of met demand, which suits the healthcare delivery application, that differs from the aforementioned papers. Kwizera and Nurre (2018) propose a two-level integrated inventory model to manage internal operations in a drone swap station delivering to multiple customers (or classes, equivalently) but exclude the uncertainty in the system. To the best of our knowledge, we are the first to introduce the stochastic SA-MCD for managing internal operations in a swap station facing stochastic demands from different geographical locations.

In recent years, there has been a rapid growth in using drones for many innovative applications (Macrina et al., 2020). Several papers (Barmounakis et al., 2016; Chang & Lee, 2018; Khosafi et al., 2019; Otto et al., 2018) review the applications of drones in different contexts, and we refer the reader to Otto et al. (2018) for an extensive review on the optimization approaches for civil applications of drones. Delivering portable medical items such as blood units and vaccines using drones can positively impact the levels of medical service in remote or congested places where roads are not a viable option for transportation and delivery (Dhote & Limbourg, 2020; Otto et al., 2018). Several companies are using drones to deliver medical supplies in different parts of the world (Chandler, 2020; Kent, 2019; Lyons, 2020; Matternet, 2020). Notably, we focus on a case study related to Zipline, a drone delivery company that started with 15 drones delivering medical items to remote locations in Rwanda in 2016 (Staedter, 2016). After successful operations in Rwanda, Zipline expanded its medical delivery service in the south of Ghana using 30 drones in 2019 (Bainbridge, 2019). During the COVID-19 pandemic, drone delivery provides a fast, cheap, and reliable method to distribute COVID-19 vaccines. For instance, in Ghana, Zipline already delivered 11000 doses and will deliver more than 2.5 million doses in 2021 (Vincent, 2021). Draganfly, a Canada-based company, will use drones to distribute COVID-19 vaccines to remote areas of Texas starting in Summer 2021 (Singh, 2021).

A drone can store a limited amount of energy that restricts its flight range. This limitation needs to be considered when modeling real-world problems. Common modeling approaches are to use the maximal operational time (Tokekar et al., 2013; Wang et al., 2017) and maximal flying distance (Guerrero et al., 2014; Savuran & Karakaya, 2016). In this paper, we use the maximal coverage of 80 km radius (160 km round-trip) (Engineering for Change, 2021) from our swap station, which is located at the Muhanga Zipline drone hub, for geographically-based demand classification in our case study.

Our SA-MCD problem belongs to the class of sequential decision-making under uncertainty problems, and we model this as a Markov Decision Process (MDP), which is appropriate for this class of problems (Puterman, 2005). There is extensive research on the use of MDPs for stochastic problems in the operations research community. A sample of problems and applications that are close to our research include drone applications (Al-Sabban et al., 2013; Baek et al., 2013; Fu et al., 2015), dynamic inventory and allocation (Federgruen & Zipkin, 1984; Somarini et al., 2017), and optimal timing of decisions (Alagöz et al., 2004; Chhatalwal et al., 2010; Khojandi et al., 2014; Zhang et al., 2012).

In SA-MCD, we use demand classification that researchers broadly utilize to study scheduling, allocation, supply chain management, and inventory control problems. We discuss a sample of scientific works that used such a classification in combination with an MDP modeling approach. Gayon et al. (2009) provide optimal production policies for a supplier facing multiple classes of demands differing in demand rates, expected due dates, cancellation probabilities, and shortage costs. Benjaafar et al. (2011) formulate an MDP model to derive optimal production policies for an assembly system wherein the demands are classified based on the difference between shortage penalties incurred due to the lack of inventory to satisfy orders. Thompson et al. (2009) categorize patients served by a hospital according to the floors treating patients and the lengths of stay in hospitals. Milnar and Chevalier (2016) use an infinite horizon MDP to model an admission control problem to maximize the expected total profit of a firm serving two classes of customers. The customers are classified based on profit margins, order sizes, and lead time. We use a geographically-based demand classification influenced by different flight ranges of a drone based on the level of charge inside its batteries. A related work considering travel range for demand classification can be found in Wang et al. (2019) wherein it is assumed that different types of EVs have different drive ranges. They accumulate the distance traveled by different users to find a total travel distance (daily deterministic demand) covered by charging stations’ deployment. They provide a long-term decision framework based on the charge required for EVs operating in a major city. We incorporate the travel range for drones in a different way, which is in line with our modeling approaches and elements. In our work, demand comes from various nodes located at different distances from one predetermined station. We have an operational-level view considering the number and timing of blood units needed in different hospitals generating uncertain demands depending on the hours of a day. Wang et al. (2019) provide a framework that estimates the charging demands for charging stations and determine the service capacity of the stations without optimizing the system. Our work differs from theirs as we focus on optimizing decisions under uncertainty.

MDP models typically suffer from the curses of dimensionality (Powell, 2011; Sutton & Barto, 2018), which means standard exact solution methods (e.g., dynamic programming Puterman, 2005) are not tractable. Hence, researchers typically resort to approximate methods, such as Reinforcement Learning (RL) or approximate dynamic programming (ADP) (the term more used in the operations research community (Powell, 2011). There is an extensive use of RL methods for solving routing, scheduling, and allocation problems and recent examples can be found in the works of Chen et al. (2021), Kool et al. (2019), Wu et al. (2021), Xin et al. (2020), and He et al. (2020). We refer the reader to Li et al. (2021) for an overview of learning-based optimization algorithms for vehicle routing problems. Examples of various ADP/RL methods in dynamic allocation problems (like ours) are temporal difference learning (Cimen & Kirkbride, 2013, 2017; Roy et al., 1997), case-based myopic RL (Jiang & Sheng, 2009), Q-Learning (Chaharsoughi et al., 2008), value function approximation (Bertsimas & Demir, 2002; Erdelyi & Topaloglu, 2010; Maxwell et al., 2010), linear function approximations (Powell & Topaloglu, 2005), and policy iteration (Nasrollahzadeh et al., 2018). In this paper, we apply a value function approximation using a lookup table (e.g., Jiang & Sheng, 2009; Kwon et al., 2008) with an -greedy exploration feature (Powell, 2011; Ryzhov et al., 2019) to make our RL method visit and update the value of more (both attractive and unattractive) states in the state space. We reduce the exploration rate over the iterations, to converge to a good value function approximation learned by a near-optimal exploitation policy. In Section 4.2, we present a comprehensive explanation of our RL method.

3. Problem description and formulation

In this section, we present our MDP approach to model the stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD). We proceed by formally describing the classes of demand and the components of our MDP model.

For our problem, we consider a set of medical facilities, each with an unknown number of requests (i.e., demand) for drone delivery over time. We know how long a drone needs to fly from the drone hub (located in the swap station) and back to satisfy a request for each medical facility. We cluster medical facilities with similar flight times
into demand classes. The demand for each medical facility is then aggregated by a demand class. The uncertainty in our MDP is given by the number of requests (i.e., demand) for each demand class over time. We assume that there is a known probability distribution that governs the uncertainty for each demand class over time. We depict an example of geographically-based demand classification in Fig. 1.

We link each demand class with the required amount of battery charge that is necessary to make the round-trip flight between the drone hub and the medical facility. In other words, higher demand classes that are farther from the drone hub require more charge than those closer to the hub. Charging all batteries to full charge ensures that each drone+battery pair can satisfy a request from any demand class. However, in reality, this strategy results in a higher total cost, longer recharge times, and faster battery degradation. On the other hand, demand classification enables the drone delivery company to have a request from class \( i \) satisfied using a battery from the lowest, capable, available charge level. For example, imagine we have a request from demand class 2. Any battery with charge level 2…\( C \) is capable of satisfying this request. If a battery is available with charge level 2, this battery is used. However, if no batteries are available with charge level 2, then we look for a battery with lowest charge level higher than 2. If none are available, we designate this demand as unmet. With this assumption, we maintain higher charge level battery in inventory.

**Actions:** We use \( a_t \) to denote the recharging action at time \( t \) using a vector of size \((C+1)/2\) such that \( a_t^k \) represents the number of batteries starting at charge level \( k \) which are recharged to level \( k \) for \( k > i \) at time \( t \). As follows,

\[
a_t = \{a_t^1 \in A_t^k : \forall i = 0, 1, \ldots, C - 1, k = i + 1, \ldots, C\}
\]

where

\[
A_t^k = \{0, 1, \ldots, s'_t\} \quad \forall i = 1, \ldots, C - 1, k = i + 1, \ldots, C, \text{ and}
\]

\[
A_t^0 = \{0, 1, \ldots, M - \sum_{i=1}^{C} s'_t\} \quad \forall k = 1, \ldots, C.
\]

To ensure that the number of batteries selected to be recharged from each charge level does not exceed the number of batteries within that class, we force the actions to satisfy Eqs. (5) and (6).

\[
\sum_{k=1}^{C} a_t^k \leq M - \sum_{i=1}^{C} s'_t,
\]

\[
\sum_{k=1}^{C} a_t^k \leq s'_t \quad \forall i = 1, \ldots, C - 1, \forall t \in T.
\]

In Fig. 2, we display the state transitions between different states due to different recharging actions or demand satisfaction for a single battery. Recharging/demand satisfaction increases/decreases the level of charge of a battery depending on the level of recharging/classes of met demand.

**Transition Probabilities:** The system transitions from state \( s_t \) to a future state \( s_{t+1} \) according to the selected action and the realized demand within each demand class. In our system, the demand at time \( t \), denoted \( D_t \), is a vector of size \( C \), i.e., \( D_t = (D_t^1 \ldots, D_t^C) \) where each \( D_t^i \) for \( i = 1, \ldots, C \), is a random variable representing the number of requests for demand class \( i \). As we explain later, the state transitions and the probability transition function are complex, but we illustrate these functions of our MDP for \( C = 2 \). However, we note that the model could be applied to a problem with \( C > 2 \).

As our state transition is complex, we first define an intermediate state of the system. We define the intermediate state of the system

\[
S = \{s'_t = (s'_t^1, \ldots, s'_t^C) \in S^1 \times S^2 \times \cdots S^C, \forall \in 1, \ldots, C\}
\]

\[
\sum_{i=1}^{C} s'_t^i \leq M, \forall t \in T
\]

Fig. 1. An example of demand classification based on the distance between the location of demand and the drone hub.
as $L = (L^1, L^2)$ and allow this to represent the number of batteries currently charged at levels 1 and 2 after all actions are taken and batteries are allocated to satisfy demand within their class (i.e., batteries with charge levels 1 and 2 are used to satisfy the demand of class 1 and class 2, respectively). Therefore, the intermediate states should not be mistaken for the so-called post-decision states that show the system’s state immediately after making decisions before realizing the uncertainty. We note, this intermediate transition does not incorporate the batteries from charge level 2 used to satisfy remaining demand from class 1. The transitions to intermediate states are governed by Eqs. (7) and (8). In Eq. (7), $\min(s_{1}^1 - a_{1}^2, D_{1}^1)$ equals the satisfied demand of class 1 using the available batteries with charge level 1. Similarly, $\min(s_{2}^1, D_{2}^1)$ denotes the satisfied demands of class 2 using batteries with charge level 2 in Eq. (8).

$$L^1 = s_{1}^1 + a_{0}^1 - a_{1}^2 - \min(s_{1}^1 - a_{1}^2, D_{1}^1).$$

(7)

$$L^2 = s_{2}^1 + a_{0}^2 + a_{1}^2 - \min(s_{2}^1, D_{2}^1).$$

(8)

Using this intermediate state, we now present the entire state transition equations. Given $L = (L^1, L^2)$, we can now use the remaining batteries with level 2 charge to satisfy any remaining demand for class 1. We present the full future state of the system with Eqs. (9) and (10).

$$s_{1}^t_{1+1} = L^1 + \min\{\max(0, D_{1}^1 - (s_{1}^t - a_{1}^2)), \max(0, s_{2}^t - D_{2}^1)\}.$$  

(9)

$$s_{2}^t_{1+1} = L^2 - \min\{\max(0, D_{1}^1 - (s_{1}^t - a_{1}^2)), \max(0, s_{2}^t - D_{2}^1)\}.$$  

(10)

To illustrate the state transitions, we use Fig. 3 to display the timing of events between the two consecutive decision epochs. At epoch $t$, we observe the state of system $(s_{1}, s_{2})$ and make a decision for charging actions $(a_{0}, a_{1}, a_{2})$. Then, we realize the stochastic demand at epoch $t$. It means the information for the number of demands of each class becomes available after we make the decisions for charging actions. Between two decision epochs, we first allocate drones with level 1 and 2 charges to satisfy the demand class 1 and 2, respectively. Then, we can determine the intermediate state of the system and allocate the leftover batteries of level 2 charge to meet the unmet demand of class 1. Ultimately, at epoch $t + 1$, we update the state using Eqs. (7), (8), (9), and (10).

In Eqs. (9) and (10), $U^1 = \max(0, D_{1}^1 - (s_{1}^t - a_{1}^2))$ is the amount of unsatisfied class 1 demand after using level 1 charged batteries and $U^2 = \max(0, s_{2}^t - D_{2}^1)$ is the number of leftover level 2 charged batteries after satisfying class 2 demands. Hence, the amount of remaining class 1 demand that can be satisfied using remaining level 2 charged batteries is the minimum of $(U^1, U^2)$. We note that our system holds the Markov property that means the system’s future state does not depend on the state of the system in the past and can be derived using solely the present state, taken action, and realized uncertainty (Puterman, 2005).

We provide an example to clarify the state transitions. Suppose we have 10 batteries and the system’s state is $(s_{1}, s_{2}) = (3, 6)$, which means 3 and 6 batteries have level 1 and 2 charges, respectively, and one battery is depleted. Let us assume, we take the action $(a_{0}^1, a_{0}^2, a_{1}^2) = (0, 1, 2)$ and the realized demand $(D_{1}^1, D_{2}^1) = (5, 2)$. Incorporating this information, the number of met demand of class 1 and 2 using batteries with the same level of charge equals $\min(s_{1}^1 - a_{1}^2, D_{1}^1) = \min(3 - 2, 5) = 1$ and $\min(s_{2}^1, D_{2}^1) = \min(6, 2) = 2$, respectively. The amount of unsatisfied class 1 demand is $U^1 = \max(0, D_{1}^1 - (s_{1}^1 - a_{1}^2)) = 4$. The number of leftover level 2 charged batteries that can be used to satisfy unmet demand of class 1 is $U^2 = \max(0, s_{2}^1 - D_{2}^1) = \max(0, 6 - 2) = 4$. Using Eqs. (7) and (8), the intermediate state of the system is $(L^1, L^2) = (0, 7)$. Then, we use 4 out of 4 level 2 charged batteries to satisfy the demand of class 1 as $\min(U^1, U^2) = 4$. These 4 batteries will return to the station with level 1 charge. Hence, the future state $s_{1}^t_{1+1} = L^1 + 4 = 4$ and $s_{2}^t_{1+1} = L^2 - 4 = 3$.

Now, we present the transition probability function from state $s_{t}$ to state $s_{t+1} = j = (j^1, j^2)$ using Eq. (11). In this equation, $p^t_{ij} = P(D_{1}^t = x)$ and $q^t_{ik} = \sum_{x \geq 1} p^t_{ix} = P(D_{2}^t \geq x) \forall i, k$. In all of the cases in Eq. (11), the intermediate state transitions are calculated using Eqs. (7) and (8). The first case in Eq. (11) calculates the transition probabilities when the stochastic demand of class 1 and 2 is less than the number of charged level 1 and 2 batteries, respectively. The future state equates the intermediate state for each charge level. In the second case, the demand of level 2 is greater than or equal to the number of batteries with level 2 charge; hence the number of batteries with level 2 charge at time $t + 1$ equals the number of recently charged batteries from empty or level 1 charge. The future state of the system equates to the intermediate state of the system. The third case is similar to the second case except that in the second case, the stochastic demand of class 1 is less than the number of level 1 charged batteries but in the third case, the demand is greater than or equal to the number of level 1 charged batteries. The fourth case describes the condition that all of the demand for the class 1 charge can be satisfied using all the available level 1 charged batteries plus the leftover batteries of level 2 after satisfying the demand of class 2. The future state of level 2 batteries will be no more than the intermediate state for level 2. The amount of satisfied demand in stage 2 (satisfying demand of class 1 using remaining batteries of class 2) equals the difference between the intermediate and future state of level 2 charged batteries. The fifth case is similar to the fourth case, except that all of the demands of
class 1 cannot be satisfied in the first and second stage of the demand satisfaction process.

\[ p(j|s_t, a_t) = \begin{cases} 
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
(p^1_{s^1, a^1 \rightarrow L^1, s^2}) (p^2_{s^1, a^2 \rightarrow L^2, s^2}) & \text{if } a^1 < L^1 \leq s^1 + a^1 - a^2, \\
0 & \text{otherwise.} 
\end{cases} \]

(11)

We note that if the number of classes increases to \( C = 3 \), then there are 13 different cases to be considered. Further, we need to consider two intermediate states to capture demand satisfaction with the same level of charge and one two-level of charge higher. For the application of our problem, two demand classes are adequate to observe a significant improvement in the quality of solutions (see Section 5) without making the model excessively complicated. If the application requires having more than three classes, we suggest using alternative strategies such as non-Markovian modeling and/or only applying ADP/RL/Q-learning methods, which does not require an explicit transition probability function.

Despite the apparent complexity of our transition probability function, we would like to point out the benefit of our MDP model. Incorporating the assumptions for the order of demand satisfaction and timing of events in our model enables us to capture the system’s dynamics without including the past state(s) information. As a result, our model holds the Markov property, allowing us to benefit from the MDP properties and guaranteed solution methods.

**Reward:** We calculate the immediate reward of taking action \( a_t \) when the transition from state \( s_t \) to state \( s_{t+1} = j \) occurs using Eq. (12).

\[ r(s_t, a_t, j) = \rho^{11} s_t^1 + a^1 - a^2 - L_1 + \rho^{21} (L_2 - j^2) + \rho^{22} (s_t^2 + a^2 - a_2^2 - L_2) \]

for \( t = 1, \ldots, N - 1 \). We use \( \rho^j \) to put weights on the amount of met demand of class \( j \) using level \( j \) charged batteries.

In the first term, \((s_t^1 + a^1 - a_2^2 - L_1)\) denotes the number of level 1 charged drones used to satisfy class 1 demand. In the second term, \((L_2 - j^2)\) equals the number of level 2 charged batteries used to satisfy class 1 demand. In the third term, \((s_t^2 + a^2 - a_2^2 - L_2)\) determines the number of level 2 charged batteries used to satisfy class 2 demand. We note that our objective is to maximize the expected total satisfied demand which does not directly incorporate cost. However, with adjusting the weights of \( \rho^j \), we implicitly include a cost factor by assigning a penalty/reward to demand satisfaction with an excessive level of charge. For instance, when \( \rho^{31} = \rho^{32} = \rho^{31} = 1 \), there is no benefit in recharging batteries to level 1 to satisfy class 1 demand because level 2 charged batteries can satisfy class 1 demand by generating the same reward while these batteries can satisfy class 2 demands, too. However, if \( \rho^{21} = 0.5 \), then we expect to recharge to use more level 1 charged batteries to satisfy class 1 demand given that \( a^1 = 2a^2 \), which means we penalize the reward of satisfying class 1 demand with level 2 charged batteries. We vary this parameter and analyze the results in Section 5. In practice, the drone delivery companies can select the value of this parameter based on their insights or preferences.

At the end of the time horizon we calculate the terminal reward. We assume that no action is taken at the end of the time horizon and that all remaining batteries can be used to satisfy future demand, and there is sufficient demand for each level. Thus, we define the terminal reward using Eq. (13).

\[ r_N(s_N) = \rho^{11} s_N^1 + \rho^{22} s_N^2 \]

(13)

We calculate the immediate expected reward \( r(s_t, a_t) \), using the immediate reward and transition probability functions given by Eq. (14),

\[ r(s_t, a_t) = \sum_{j \in S} \left[ p_j (s_t, a_t) \rho^{11} (s_t^1 + a^1 - a^2 - L_1) + \rho^{21} (L_2 - j^2) + \rho^{22} (s_t^2 + a^2 - a_2^2 - L_2) \right] \]

(14)

We derive the decision rules, \( d_t(s_t) : s_t \rightarrow A_{s_t} \), from the action set to maximize the total expected reward. Selecting an action when in state \( s_t \) and time \( t \) is equivalent to giving value to the associated decision variable. Because we select a single action based on the present state, which does not depend on the past states and actions, our decision rules belong to the Markovian decision rules (Puterman, 2005). A policy \( \pi \) is a sequence of decision rules for all decision epochs, that is \( d_t(s_t) \forall t \in \)}
T. We can calculate the expected total reward of policy $\pi$ for the problems starting from an arbitrary initial state $s_1$ using Eq. (15). The optimal policy, $\pi^*$, maximizes the expected total reward.

$$V_N^\pi(s_1) = E_N^n \sum_{t=1}^{N-1} r_t(s_t, a_t) + r_N(s_N)$$  \hspace{1cm} \text{(15)}$$

### 4. Solution methodology

In this section, we first present the exact solution method, backward induction (BI), and continue with our approximate solution method, the reinforcement learning method with a descending $\epsilon$-greedy exploration feature (DeRL) to solve the stochastic SA-MCD. We note that RL is a powerful method to provide approximate solutions for large-scale MDPs (Powell, 2011; Sutton & Barto, 2018). We proceed by introducing the notation in Table 1 and continue with our solution methods.

#### 4.1. Backward induction

As our Markov Decision Process (MDP) model has finite state and action spaces, there is at least one deterministic optimal policy (Puterman, 2005). Backward induction (BI) can determine the policy (number of recharging actions) that maximizes the expected total reward or weighted demand over time. Let $V_N^\pi(s_t)$ be the optimal value function equivalent to the maximum expected total reward from decision $t$ onward when the system is in state $s_t$. Then, we can use the optimality equations, given by Eq. (16), to find the optimal policies for all the decision epochs when moving backward in time. That is, BI sets the value of being in state $s_N$ at the end of the time horizon $N$ to be equal to the terminal reward value given by Eq. (13). Then, the algorithm starts from the last decision epoch and finds the optimal actions ($a_{n,t}^*$) and corresponding values ($V_n^* (s_t)$) using Eqs. (16) and (17) stepping backward in time. The algorithm aims to find the optimal expected total reward over the time horizon, $V_N^\pi(s_1)$, for state $s_1$, which is the system’s initial state at time $t = 1$. In other words, solving the optimality equations for $t = 1$ is equivalent to the expected total reward over the time horizon.

$$V_n^\pi(s_t) = \max_{a_t \epsilon A_t} \left\{ r_t(s_t, a_t) + \sum_{j \epsilon S} p_t(j \mid s_t, a_t) V_{n+1}(j) \right\} \hspace{1cm} \text{(16)}$$

$$a_{n,t}^* = \arg \max_{a_t \epsilon A_t} \left\{ r_t(s_t, a_t) + \sum_{j \epsilon S} p_t(j \mid s_t, a_t) V_{n+1}(j) \right\} \hspace{1cm} \text{(17)}$$

The size of state space, action space, transition probability, and optimal policies (the best actions for all the states over time) are functions of $O(M^3)$, $O(M^2)$, $O(M^3 N)$, and $O(M^2 N)$, respectively. The complexity of BI is $O(M^2 N)$. As the size of the problem increases, it becomes challenging for BI to find the optimal solution due to the curses of dimensionality, which causes a drastic increase in computational time and memory. Hence, we proceed with presenting our reinforcement learning (RL) method, which is capable of circumventing such problems (Powell, 2011; Sutton & Barto, 2018). Our RL method has a significantly lower complexity of $O(M^3 N \tau_1)$ wherein $\tau_1$ is the number of iterations. As shown in Section 5, the number of iterations with an order of $10^8$ is sufficient to provide a tight optimality gap for different problem sizes.

#### 4.2. Reinforcement learning

In this section, we explain the reinforcement learning (RL) method to find near-optimal policies and to cope with the curses of dimensionality (Powell, 2011) of the stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD). In general, the RL methods approximate the value functions or policies (Powell, 2011; Sutton & Barto, 2018). There are many combinations of actions over the time horizon for our problem, so it is not trivial to iteratively generate a feasible policy in an improving direction. Thus, approximate policy iteration methods that require creating feasible policies are less preferred than the approximate value iteration methods. We use a value iteration based reinforcement learning with a descending $\epsilon$-greedy exploration feature (DeRL). The exploration feature helps to avoid getting trapped in local optima and thoroughly searches the action space. Our value function approximation uses the lookup table to store the value of states and the selected action per state over time; therefore, encoding and decoding, which is typically used for deep RL, is not required. The RL method is in-line with our modeling approach and provides high-quality approximate solutions for the SA-MCD (see Section 5).

#### Algorithm 1 Reinforcement Learning Method with a Descending $\epsilon$-greedy Exploration Feature (DeRL)

1: Initialize $M$ drones, $N - 1$ decision epochs, $\tau_1$ iterations, and $\tau_2$ sample paths
2: Set $V_N(s) = r_N(s)$ for $s \in S$ and $n = 1, \ldots, \tau_1$
3: Set $n = 1$
4: while $n \leq \tau_1$ do
5: \hspace{1cm} Select initial state $s_1$
6: \hspace{1cm} for $t = 1, \ldots, N - 1$ do
7: \hspace{2cm} Generate a random number $u$
8: \hspace{2cm} if $\text{Rand} < \epsilon^*$ then
9: \hspace{3cm} Sample an observation of the uncertainty, $D_t$
10: \hspace{3cm} Determine a random feasible action, $a_t^*$
11: \hspace{3cm} Calculate the observed value, $V_t(s_t) = V_t(s_t)$
12: \hspace{2cm} else
13: \hspace{3cm} Generate $\tau_2$ sample paths of the uncertainty
14: \hspace{3cm} Select an action that maximizes $V_t(s_t) + \alpha$ over $\tau_2$ sample paths
15: \hspace{3cm} Sample an observation of the uncertainty, $D_t$
16: \hspace{3cm} Calculate the observed value, $V_t(s_t)$
17: \hspace{2cm} end if
18: \hspace{1cm} Smooth the new observation with the previous approximated value,
19: \hspace{1cm} $V_t(s_t) = (1 - \alpha) V_{t-1}(s_t) + \alpha V_t(s_t)$ \hspace{1cm} \text{(18)}
20: \hspace{1cm} Update the present approximation using the smoothed value,
21: \hspace{1cm} $V_t(s_t) = V_t(s_t)$
22: \hspace{1cm} Determine next state, $s_{t+1}$
23: end for
24: Increment $n = n + 1$
25: end while

We first determine the number of drones ($M$), decision epochs ($N - 1$), ($\tau_1$) iterations, and ($\tau_2$) sample paths (of realized uncertainty, visited states, and policies per time) in our RL method. Then, we initialize the approximate value at the end of the time horizon, $t = N$, for all iterations using the terminal reward function given by Eq. (13). For
every iteration, we select an initial state, $s_t^0$. To select the action, we use the ε-greedy method (Powell, 2011) that allows exploring the action space and works as follows. We generate a random number $\epsilon$. Then, we compare $\epsilon$ with the exploration rate, $\epsilon^*$, at iteration $t$. We use a descending function for the exploration rate over the iterations to ensure more states (both attractive and unattractive) are visited and facilitate the algorithm convergence. If $\epsilon < \epsilon^*$, we select a feasible action arbitrarily. Otherwise, we generate $r_t$ sample paths of demands (realized uncertainty) and select the action that maximizes the observed value $\tilde{V}_t(s_{t+1})$ according to Eqs. (19) and (20), wherein $\tilde{V}_t(s_{t+1})$ is used to approximate the value of $E[V_{t+1}(s_{t+1}) \mid s_t, a_t]$ for each sample path. If an action is selected over multiple sample paths, we use the average of $\tilde{V}_{t+1}(s_{t+1})$ as the approximation. The observed value and the approximated value at the previous iteration are smoothed using a stepsize function. This value is now used as the present approximation value of the observed state. When an action is selected, we sample an observation of uncertainty (generate a realized value for stochastic demand) to find the future state. The algorithm steps forward in time and moves to the future observed state until it reaches the last decision epoch and a new iteration starts. The same process is repeated until $r_t$ iterations are completed.

\[
\text{a}_{t+1} = \arg \max_{a_t \in A_t} \left\{ r_t(s_t, a_t) + \epsilon^{r_t}(s_{t+1}) \right\}. \tag{19}
\]

\[
\tilde{V}_t(s_{t+1}) = \max_{a_t \in A_t} \left\{ r_t(s_t, a_t) + E[V_{t+1}(s_{t+1}) \mid s_t, a_t] \right\}. \tag{20}
\]

5. Computational results

In this section, we explain the results of solving the stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD) using realistic data related to the drone delivery company Zipline. We created this dataset to mimic the geographical locations of the Zipline drone hub and hospitals in Rwanda, Africa, the population of districts, flight regulations in the country, Zipline drone configuration, including the speed, flight range, and recharging time. We solve modest SA-MCD (15–21 drones) using exact solution methods. Note that we use the number of batteries and the number of drones interchangeably as each drone has a battery pack, which might consist of multiple batteries that can be charged in parallel. As we run into the curses of dimensionality for larger instances of SA-MCD, we present the results of our reinforcement learning (RL) method that can provide near-optimal solutions for the modest instances and solve larger problem instances. We deduce managerial insights for managing the swap station’s distribution operations that maximize the expected total weighted met demand of multiple classes. Finally, a set of experiments are performed to more generally validate the performance of our RL method on a fictitious case considering the location of hospitals in a different region. Here, many random scenarios are generated based on possible values for the requested flights from hospitals and the objective function multiplier, $\rho^{21}$. The results show the robustness of our RL method in providing high quality solutions in terms of the optimality gaps and average percentage of met demands. We proceed by first explaining the data in Section 5.1.

5.1. Data

In this paper, we present an actual case study related to the vital operations of Zipline drones, which are delivering medical items in Rwanda, Africa. The Zipline station is located at the Muhanga district, west of Rwanda’s capital city, Kigali. We focus on drone delivery to satisfy the stochastic demand for blood units originating from hospitals in Rwanda. In Table 2, we summarize the input data, including, the name of each hospital, their location, the distance between the station and each hospital, the approximated population of people using each hospital, the number of blood units and flights needed per day for
Fig. 4. Locations of hospitals (demand nodes), the swap station located in the Zipline drone hub, and airports in Rwanda.

each hospital, and the demand class associated with each hospital. We categorize the demand into two classes based on the distance between the Zipline station and hospitals. As the flight range of the drone is estimated to be 80 km (Ackerman, 2020), we assume demands of hospitals located within [0 km, 40 km] and [40 km, 80 km] fall into class 1 and class 2, respectively. The demand class NA means that the hospital is not reachable from the Zipline station and excluded from further analysis. We exclude six hospitals (denoted by NA in the last column of Table 2) from the 33 identified hospitals as they are located at a distance out of the drone’s flight range from the Zipline station. We consider 10 hospitals in class 1 and 17 hospitals in class 2, which are located throughout 15 distinct districts in Rwanda (see Table 2).

We approximate the air travel distance between the station and hospitals using the Haversine formula (Sinnott, 1984) that is broadly used to find the distance between two points on the earth. When calculating the distance, we consider the rules for flying drones in Rwanda, which does not allow drones to fly within a 10 km radius from airports (Rwanda Civil Aviation Authority, 2021). Therefore, we need to adjust the travel distance between the station and two hospitals, Kiziguro and Rwamagana. Hence, we calculate the closest travel distances such that the flights to these destinations do not violate the rules for flying drones in Rwanda. In Fig. 4, we display the geographical locations of airports, hospitals, and the Zipline station.

Consistent with Swartzman (1970) and Armony et al. (2015), we use a non-homogeneous Poisson process to determine the patients’ arrival to the hospitals. We examine the daily operations of the drone swap station wherein the time between two consecutive decision epochs is 90 min (1.5hr). Thus, $N = \frac{24}{1.5} + 1 = 17$. The 90-minute intervals provide adequate time for drones to receive charge (McNabb, 2020) and complete a round-trip from the furthest delivery mission to the station given that the maximum speed of the drone is 127 km/hr (Petrova & Kolodny, 2018).

To derive the mean demand of blood units per time $t$, we apply the following process. First, we determine the number of people using a particular hospital based on the population of each district. If more than one hospital is located in a district, we evenly distribute the district’s total population over the number of hospitals located in that district. Second, we calculate the number of blood units needed per year for each hospital by multiplying an estimated portion of the population that needs blood units per year (2% recommended by the World Health Organization (WHO) (Dhingra, 2010)) by the number of people using that hospital. The resulting number is an overestimate of the number reported by Iliza (2020). However, we use this number to account for pessimistic cases wherein the station faces more demand. Third, we divide the number of blood units required per year by 365 to find the number of blood units needed per day for each hospital. Next, we use the pattern of patient arrivals to hospitals, consistent with Green et al. (2007), Tiwari et al. (2014), and Jones et al. (2007), to derive the mean demand for blood units of time $t$ over a day. This pattern shows an ascending trend of arrivals from 6:00am to the peak at noon, followed by a descending trend from noon to 6:00am of the following day. Specifically, we use the data from Green et al. (2007) and fit a polynomial function for generating the mean arrival rate at time $t$. In Fig. 5, we display the mean demand of Green et al. (2007) and our fitted function. We scale the mean demand of blood units of time $t$ such that the summation of the scaled demand over a day equals the calculated number of blood units required per day for each hospital. Then, as each drone has a fixed load capacity and can carry two units of blood (Baker, 2017), we divide the mean demand of blood units of time $t$ by two to find the mean demand for flights for each hospital. We note that if the number of required blood units, and, in turn, the number of requests for drone flights is more than the number of available drones in the station, the demand is lost. We do not allow backlogging or rolling over unmet demands because the mission of our drones is delivering medical items, which are often vital at the moment of orders. Finally, the mean demand for either demand class, $\lambda_1(t), \lambda_2(t)$, is the aggregation of mean demands for flights from the hospitals within each demand class.
In the first experiment, we consider Zipline has a fleet of 15 drones (Staedter, 2016). We set $p^{11} = p^{22} = 1$ and $p^{11} = 0.5$ indicating that satisfying a demand of class 1 using a level 1 charged battery (partially-charged) generates more immediate reward than satisfying that demand using a level 2 charged battery (fully-charged). This setting implies the company provides less reward when drones with excessive level of charge are used to satisfy demands, which can be interpreted as a penalty to account for unnecessary higher recharging costs incurred.

Next, the selection of the hyperparameters of DRL is explained. The number of core iterations is $r_1 = 200000$. As we will see later, the algorithm will converge after 50000 iterations; however, as the computational time is in a matter of minutes, we keep 200000 iterations for our experiments. We test $r_2 = 1, 5, 10, \ldots, 50$ and observe that increasing the parameter to the value of 30 reduces the optimality gap and increases the robustness of the results and computational time, but excessive increase in the value of $r_2$ only magnifies the computational time with little improvement in the quality of the result; thus, we set $r_2 = 30$. We use the adaptive stepsize function provided by George and Powell (2006). We use $\epsilon^n = \frac{1}{n}$ to adjust the value of the exploration rate at iteration $n$ used in the $\epsilon$-greedy approach to select policies within our RL method. With this function, we ensure a higher rate of exploration/exploitation in early/late iterations, which is desirable for visiting more states and enabling the algorithm to converge as it proceeds with each iteration.

5.2. Analysis and discussion

In this section, we first feed the data explained in Section 5.1 to solve the problem using exact and approximate solution methods, Backward Induction (BI) and the reinforcement learning method with a descending $\epsilon$-greedy exploration feature (DRL), respectively. Then, we analyze the optimal policies (BI solutions) and assess the quality of near-optimal solutions derived from DRL. Moreover, as the drone delivery company can control and adjust $p^{11}$ (the weight of satisfying class 1 demand using level 2 charged batteries), we analyze the impact of changing the parameter’s value on the station’s operations and amount of met demand. We also conduct different sets of experiments to solve various instances of the problem and answer the following questions: How many batteries are needed in the station to satisfy a certain level of the stochastic demand? What is the contribution of classifying the demand on the demand satisfaction? We use a high-performance computer with four shared memory quad Xeon octa-core 2.4 GHz E5-4640 processors and 768 GB of memory for running all of our computational tests.

5.2.1. Comparing results of BI and RL

In this section, we present the results from solving the stochastic SA-MCD with BI and DRL and using the data presented in Section 5.1. The system’s initial state is $s_1 = (0, 15)$, which means all 15 batteries are charged to level 2 (fully-charged). The time horizon is one day and $N = 17$ wherein the first decision epoch is at midnight and the time between any two consecutive decision epochs is 90 min. That is, the decisions are made at 16 decision epochs, $t_i$ where $t = 0:00, 1:30, 3:00, \ldots, 10:30, 12:00, 13:30, \ldots, 22:30$. The learning phase of our RL method is finished in iteration $t_1 - 1$, and the last iteration ($t_N$) is used to evaluate the method’s performance. The results of DRL in the last iteration can differ in terms of the converged value and met demand. Hence, we generate independent sample paths to report robust results and evaluate our RL method’s performance. Our preliminary experiments reveal that our result is very robust when more than 100 sample paths of demand are incorporated for evaluation (the percentage change in the converged values is satisfactory and less than 0.1). Hence, to yield even a higher robustness level, we calculate the average percentage of met demand using Eqs. (21) and (22) wherein we generate 500 sample paths of realized demand using the Poisson distribution at time $t$ for each class of demand.

\[
\text{\% of Demand Met over Time for a Sample Path} = \frac{\text{Tot. # Met Dem. over Time} - \text{Tot. # Realized Dem. over Time}}{\text{Tot. # Realized Dem. over Time}} \times 100\%.
\]

\[
\text{Average (\%) of Demand Met} = \frac{\sum_{i=1}^{N} \% \text{ of Met Demand over Time for Sample Path } i}{\text{# of Sample Paths}}.
\]

We summarize the results in Table 3. The percentage of met demand over a sample path equals the total number of demand met over the total realized demand of both classes. We report the average of 500 sample paths in Table 3. We provide more detailed results about demand satisfaction by class later in Table 5. As shown, DRL is faster and can generate a high-quality solution with 5.3% of optimality gap (derived from Eq. (23)) in 8 min. We note that the reported computational time consists of both learning and evaluation phases. When the policy is learned from the learning phase, it takes seconds. The computational time for evaluating the policy of the RL method is shown in Table 3. We provide more detailed results about demand satisfaction by class later in Table 5. As shown, DRL is faster and can generate a high-quality solution with 5.3% of optimality gap (derived from Eq. (23)) in 8 min. We note that the reported computational time consists of both learning and evaluation phases. When the policy is learned from the learning phase, it takes seconds.

Table 3

<table>
<thead>
<tr>
<th>Solution method</th>
<th>Expected total reward</th>
<th>Average met demand (%)</th>
<th>Computational time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI</td>
<td>115.1</td>
<td>63.7</td>
<td>6740.7</td>
</tr>
<tr>
<td>DRL</td>
<td>109.0</td>
<td>60.9</td>
<td>483.7</td>
</tr>
<tr>
<td>Benchmark</td>
<td>105.6</td>
<td>58.5</td>
<td>6442.1</td>
</tr>
</tbody>
</table>

In Table 3, we also provide the results of an easy-to-implement intuitive benchmark. As the model’s objective is to maximize the amount of met demand, our proposed benchmark policy makes all the empty batteries fully-charged (usable for demand satisfaction of all classes) at every decision epoch. We note the benchmark can be quickly implemented, but it needs a considerable computational time to calculate the exact expected total reward. The computational time for evaluating the benchmark includes all of the operations of backward induction, except
Given the initial state, taken actions, realized demand, and the state visited states and policies) using the sample paths of realized demands.

For finding the best actions as the actions are derived from the (fixed) benchmark policy. Our results show that DRL outperforms the benchmark policy regarding the expected total reward, average demand met, and computational time. Thus, we only continue with DRL as the superior approximate solution method.

In the preliminary tests, we observe that using a descending ϵ-greedy exploration feature within the RL method can improve the quality of the solution. To test the impact of this exploration feature, we conduct a set of experiments with $M = 15, \ldots, 21$ and $\rho = 0.5$ and compare the results with RL with constant/no exploration feature. On average, the average optimality gap of RL with descending, constant, and no exploration features are 4.0%, 9.6%, 15.8%, respectively. The results highlight the importance of a declining exploration rate. The observed optimality gaps for the same set of values for $M$ with $\rho = 1.6, 1.7, \ldots, 2.0$ show similar performance for the different exploration settings.

In Table 4, we provide the summary of the results from solving the problem with 15 to 21 drones using BI and DRL. Note that BI cannot find the optimal solution when the number of drones is greater than 21. For 15–21 drones, DRL provides an optimality gap of less than 5% for all instances and significantly reduces computational time. The maximum difference between the average percentage of met demand of DRL and BI is less than 5%. Further, we again expand our experiments for $M = 15, \ldots, 21$ with $\rho = 0.6, 0.7, \ldots, 2$. The results show that the average and maximum optimality gaps are 4.2% and 5.4%, respectively, over these 7 × 15 = 105 experiments. The results indicate the high quality of the solution. To test the impact of this exploration feature, we conduct a set of experiments with $M = 15$.

A significant finding is that 15 drones are not sufficient to satisfy the demand of either class. We proceed with analyzing the impact of increasing the number of drones in the station on the amount of met demand.

### 5.2.2. Analysis of the number of required drones

In this section, we solve the problem for a larger number of drones in the station to find the relationship between this number and the amount of met demand. The analysis provides significant insights for drone delivery companies given the high price to purchase and maintain drones in swap stations. First, we note that backward induction (BI) can solve the problem with at most 21 drones using our computational resources. We summarized the amount of met demand, computational time, and memory used to solve the problem for 15 to 21 drones in Table 4 using BI and RL. For $M > 21$, we report the results of our DRL method in Table 6.

As shown, when $M \geq 54$, the average percentage of met demand over time for 500 sample paths is 100%. We depict a sample path of policies for $M = 54$ when demand equals the mean demand in Fig. 8. As all batteries are initially available with a level 2 charge, we recharge fewer drones in the early morning. Then, we recharge more batteries.

---

**Table 4**

<table>
<thead>
<tr>
<th>$M$</th>
<th>BI</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Backward induction (BI)</td>
<td>Reinforcement learning (DRL)</td>
</tr>
<tr>
<td></td>
<td>Average met demand (%)</td>
<td>Average met demand (%)</td>
</tr>
<tr>
<td>15</td>
<td>63.7</td>
<td>6740.7</td>
</tr>
<tr>
<td>16</td>
<td>66.9</td>
<td>11630.4</td>
</tr>
<tr>
<td>17</td>
<td>69.9</td>
<td>19356.4</td>
</tr>
<tr>
<td>18</td>
<td>72.6</td>
<td>31405.8</td>
</tr>
<tr>
<td>19</td>
<td>75.4</td>
<td>49812.5</td>
</tr>
<tr>
<td>20</td>
<td>77.9</td>
<td>77170.6</td>
</tr>
<tr>
<td>21</td>
<td>80.2</td>
<td>117154.0</td>
</tr>
</tbody>
</table>

**Fig. 6.** Expected total reward convergence of DRL.
Table 5

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Avg met demand C1 (%)</th>
<th>Avg met demand C1(%)</th>
<th>Avg met demand C2(%)</th>
<th>Avg met both class (%)</th>
<th>Avg $a_1$</th>
<th>Avg $a_2$</th>
<th>Avg $a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>30.0</td>
<td>13.8</td>
<td>43.6</td>
<td>74.5</td>
<td>62.6</td>
<td>0.67</td>
<td>4.97</td>
</tr>
<tr>
<td>0.6</td>
<td>25.7</td>
<td>16.2</td>
<td>41.3</td>
<td>76.2</td>
<td>64.2</td>
<td>0.45</td>
<td>5.14</td>
</tr>
<tr>
<td>0.7</td>
<td>23.8</td>
<td>17.6</td>
<td>41.4</td>
<td>76.8</td>
<td>64.4</td>
<td>0.35</td>
<td>5.21</td>
</tr>
<tr>
<td>0.8</td>
<td>22.3</td>
<td>18.9</td>
<td>41.3</td>
<td>77.1</td>
<td>64.5</td>
<td>0.27</td>
<td>5.25</td>
</tr>
<tr>
<td>0.9</td>
<td>21.0</td>
<td>20.0</td>
<td>41.1</td>
<td>77.1</td>
<td>64.5</td>
<td>0.20</td>
<td>5.29</td>
</tr>
<tr>
<td>1.0</td>
<td>15.9</td>
<td>24.9</td>
<td>40.9</td>
<td>76.9</td>
<td>64.4</td>
<td>0.00</td>
<td>5.32</td>
</tr>
<tr>
<td>1.1</td>
<td>13.9</td>
<td>27.0</td>
<td>41.9</td>
<td>77.5</td>
<td>64.6</td>
<td>0.00</td>
<td>5.24</td>
</tr>
<tr>
<td>1.2</td>
<td>13.1</td>
<td>27.6</td>
<td>40.7</td>
<td>77.5</td>
<td>64.5</td>
<td>0.00</td>
<td>5.22</td>
</tr>
<tr>
<td>1.3</td>
<td>12.6</td>
<td>28.0</td>
<td>40.7</td>
<td>77.5</td>
<td>64.5</td>
<td>0.00</td>
<td>5.20</td>
</tr>
<tr>
<td>1.4</td>
<td>12.2</td>
<td>28.5</td>
<td>40.7</td>
<td>77.3</td>
<td>64.4</td>
<td>0.00</td>
<td>5.17</td>
</tr>
<tr>
<td>1.5</td>
<td>11.5</td>
<td>29.7</td>
<td>41.3</td>
<td>76.6</td>
<td>64.1</td>
<td>0.02</td>
<td>5.08</td>
</tr>
<tr>
<td>1.6</td>
<td>12.4</td>
<td>33.4</td>
<td>46.8</td>
<td>73.4</td>
<td>63.7</td>
<td>0.15</td>
<td>4.76</td>
</tr>
<tr>
<td>1.7</td>
<td>12.5</td>
<td>36.4</td>
<td>48.9</td>
<td>70.4</td>
<td>62.9</td>
<td>0.25</td>
<td>4.48</td>
</tr>
<tr>
<td>1.8</td>
<td>11.1</td>
<td>38.1</td>
<td>49.2</td>
<td>69.7</td>
<td>62.5</td>
<td>0.28</td>
<td>4.34</td>
</tr>
<tr>
<td>1.9</td>
<td>10.3</td>
<td>39.2</td>
<td>49.5</td>
<td>69.0</td>
<td>62.1</td>
<td>0.31</td>
<td>4.24</td>
</tr>
<tr>
<td>2.0</td>
<td>9.1</td>
<td>40.4</td>
<td>49.5</td>
<td>68.6</td>
<td>61.9</td>
<td>0.32</td>
<td>4.15</td>
</tr>
</tbody>
</table>

Fig. 7. Sample paths of states (I, IV, VII), optimal policies (II, V, VIII), and demands (III, VI, IX) for $\rho = 0.5, 1, 2$ when the realized demand of either class equal mean demand.

from 6:00 to 18:00 as the demand of either class increases. Overall, more batteries are recharged to level 2 to satisfy the demand of either class.

5.2.3. Demand classification contribution

In this section, we compare the outputs of the models with and without demand classification to illustrate the contribution of classifying the stochastic demand. We focus on demand satisfaction as the crucial metric to assess the station’s success in delivering medical supplies. We provide this metric for a different number of drones, which is important for decision-makers given the high price of purchasing and maintaining drones.

In Fig. 9, we show the average percentage of met demand for a different number of drones when we use/do not use demand classification. The red color shows the percentage for the different number of drones when demand is not classified. In this model, the state of charge of batteries is either full or empty, and full batteries are used to satisfy the demand without considering the classified distance between the station and hospitals. The system’s state is the number of fully-charged batteries. The action is the number of recharging actions that make an empty battery fully-charged. The uncertainty is the stochastic demand disregarding the distance between the station and the hospitals (demand nodes). Hence, when demand is not classified, drones that return from (even a close) delivery mission are not available before recharging.
Table 6

<table>
<thead>
<tr>
<th>$M$</th>
<th>Average met demand (%)</th>
<th>Computational time (s)</th>
<th>Memory used (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>78.3</td>
<td>599.2</td>
<td>13.4</td>
</tr>
<tr>
<td>22</td>
<td>81.0</td>
<td>690.1</td>
<td>13.4</td>
</tr>
<tr>
<td>23</td>
<td>82.0</td>
<td>781.4</td>
<td>13.4</td>
</tr>
<tr>
<td>24</td>
<td>86.7</td>
<td>935.2</td>
<td>13.4</td>
</tr>
<tr>
<td>25</td>
<td>88.5</td>
<td>1036.8</td>
<td>13.4</td>
</tr>
<tr>
<td>26</td>
<td>91.2</td>
<td>1037.8</td>
<td>13.4</td>
</tr>
<tr>
<td>27</td>
<td>91.8</td>
<td>1298.3</td>
<td>13.4</td>
</tr>
<tr>
<td>28</td>
<td>92.4</td>
<td>1337.3</td>
<td>13.4</td>
</tr>
<tr>
<td>29</td>
<td>94.9</td>
<td>1374.1</td>
<td>13.4</td>
</tr>
<tr>
<td>30</td>
<td>94.9</td>
<td>1384.5</td>
<td>13.4</td>
</tr>
<tr>
<td>31</td>
<td>95.6</td>
<td>1406.9</td>
<td>13.4</td>
</tr>
<tr>
<td>32</td>
<td>95.7</td>
<td>1426.6</td>
<td>13.4</td>
</tr>
<tr>
<td>33</td>
<td>95.8</td>
<td>1453.5</td>
<td>13.4</td>
</tr>
<tr>
<td>34</td>
<td>95.9</td>
<td>1633.4</td>
<td>13.4</td>
</tr>
<tr>
<td>35</td>
<td>96.1</td>
<td>1651.4</td>
<td>13.4</td>
</tr>
</tbody>
</table>

Fig. 8. Sample paths of states, optimal policies, and demands for 54 drones, $\rho^{21} = 0.5$ when the realized demand of either class equal mean demand.

Table 7

<table>
<thead>
<tr>
<th>$M$</th>
<th>Average met demand (%)</th>
<th>Optimality gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>52.3</td>
<td>49.3</td>
</tr>
<tr>
<td>16</td>
<td>55.6</td>
<td>51.2</td>
</tr>
<tr>
<td>17</td>
<td>58.4</td>
<td>54.4</td>
</tr>
<tr>
<td>18</td>
<td>63.6</td>
<td>59.8</td>
</tr>
<tr>
<td>19</td>
<td>65.1</td>
<td>62.4</td>
</tr>
<tr>
<td>20</td>
<td>67.4</td>
<td>64.6</td>
</tr>
<tr>
<td>21</td>
<td>70.3</td>
<td>67.5</td>
</tr>
</tbody>
</table>

5.3. Validation on a fictitious case

In this section, we conduct a set of experiments on a different case to more generally validate the performance of our reinforcement learning method. For this case, the actual location of 52 hospitals in the Netherlands is gathered (Buter, 2019). We assign a drone hub to a coordinate that is reachable from all hospitals (see Appendix A). According to the distance between the drone hub and hospitals, we find 12 hospitals in class 1 ($\leq 40$ km from the drone hub) and 40 hospitals in class 2 ($>40$ km and $\leq 80$ km from the drone hub).

Next, we perform many experiments to measure the performance of DRL. We generate 100 scenarios such that a random number between 5 and 15 is the total number of flights needed per day to each hospital, and $\rho^{21}$ of each scenario is a random number between 0.5 and 2. Then, we use the fitted polynomial function (see Section 5.1) for generating the mean arrival rates to each hospital at time $t$. Subsequently, we scale the outputs such that the sum of mean arrivals over a day divided by two (each flight can carry two demanded blood units) equals the total number of flights needed per day to each hospital. Then, we calculate $\lambda_1^t$ and $\lambda_2^t$ similar to Section 5.1. Finally, we solve all scenarios using backward induction, DRL, and the benchmark (see Section 5.2) for 15–21 drones where the parameters and setting of DRL are the same.
as Section 5.1. We summarize the results of 700 experiments in Table 7, which are the average of 100 scenarios per different number of drones. We note that the average percentage of met demand is reported based on 500 sample paths for each scenario and number of drones.

Now, we can compare the randomly generated scenarios (for the Netherlands hospitals) and the case study results in terms of the optimality gaps and met demands. As shown in Table 7, the average optimality gaps of random scenarios for different numbers of drones using DcRL remain under 6%, which is close to the case study results. Moreover, over the random scenarios, we observe that the average percentage of met demand of DcRL is only 4.4% less than BI at maximum, which is in line with the case study results. In addition, we compare these results with the benchmark’s performance and observe that DcRL outperforms the benchmark solution in terms of both optimality gaps and average met demands. Thus, we conclude that our DcRL method is robust and performs well in both the randomly generated experiments and our case study.

6. Conclusion

In this research, we developed the stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD) where the stochastic demand is classified based on the distances a drone can fly, which is linked to the level of charge inside the drone’s battery. The SA-MCD is used to manage distribution operations of a drone swap station located at a drone hub to maximize the amount of stochastic met demand for flights delivering medical supplies in Rwanda, Africa. We formulated the problem as a Markov Decision Process (MDP) model wherein the optimal policies determine the number of recharging batteries from one level to a higher level of charge over time when encountering stochastic demand from different demand classes. We solved the problem using backward induction (BI) and observed that we run into time/memory issues when the number of drones is greater than 21. Hence, we applied a reinforcement learning method with a descending ε-greedy exploration feature (DcRL) to find high-quality approximate solutions quickly. We designed a set of experiments to show the high performance of our DcRL method and obtain insights about how to manage the operations in the station to maximize the expected total weighted met demand when the model parameters vary. Finally, we more generally validated the performance of the DcRL method by considering and analyzing the results of a fictitious case.

We found plenty of directions and opportunities related to this work for future research. For instance, we did not consider the difference between the length of time for different charging levels (still, our model outperformed the model with no demand classification). Future research should consider the time difference between different recharging actions to capture the system’s behavior more realistically. Further, it is interesting to study varying elements such as speed and weather conditions associated with various drone missions. As our transition probability function is large-scale and complex, it is worth investigating various model-free approaches, like RL/ADP/Q-learning, that can circumvent the complexities of the burdensome function, particularly when more than three classes are needed. Hence, this research, which introduces a new set of problems that can be used in many applications, opens the avenue for future studies from many perspectives. In terms of modeling and application, we can have multiple demand classification criteria, such as level of emergency (plus distance). Future research can add backlogging unsatisfied demands if it suits the application. Additionally, future research should consider how the operational charging and use actions for different demand classes impacts battery degradation wherein excessive charging should be avoided to lead to longer battery lifecycles.

CRediT authorship contribution statement

Amin Asadi: Methodology, Conceptualization, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – reviewing and editing, Visualization, Project administration. Sarah Nurre Pinkley: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. Martijn Mes: Conceptualization, Validation, Writing – review & editing.

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.

Acknowledgments

This research is supported by the Arkansas High Performance Computing Center which is funded through multiple National Science Foundation grants and the Arkansas Economic Development Commission.

Appendix A

See Table A.8.
References


A. Asadi et al.


UNICEF Supply Division (2020). How drones can be used to combat COVID-19: Technical Report, UNICEF.


