Scheduling Allocation and Inventory Replenishment Problems Under Uncertainty: Applications in Managing Electric Vehicle and Drone Battery Swap Stations

Amin Asadi
University of Arkansas, Fayetteville

Scheduling Allocation and Inventory Replenishment Problems Under Uncertainty: Applications in Managing Electric Vehicle and Drone Battery Swap Stations

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Industrial Engineering

by

Amin Asadi
Bu-Ali Sina University
Bachelor of Science in Industrial Engineering, 2009
Iran University of Science and Technology
Master of Science in Industrial Engineering, 2013

July 2021
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Sarah Nurre Pinkley, Ph.D.
Dissertation Director

Kelly Sullivan, Ph.D.
Committee Member

Shengfan Zhang, Ph.D.
Committee Member

Mohammad Marufuzzaman, Ph.D.
Committee Member

Abstract

In this dissertation, motivated by electric vehicle (EV) and drone application growth, we propose novel optimization problems and solution techniques for managing the operations at EV and drone battery swap stations. In Chapter 2, we introduce a novel class of stochastic scheduling allocation and inventory replenishment problems (SAIRP), which determines the recharging, discharging, and replacement decisions at a swap station over time to maximize the expected total profit. We use Markov Decision Process (MDP) to model SAIRPs facing uncertain demands, varying costs, and battery degradation. Considering battery degradation is crucial as it relaxes the assumption that charging/discharging batteries do not deteriorate their quality (capacity). Besides, it ensures customers receive high-quality batteries as we prevent recharging/discharging and swapping when the average capacity of batteries is lower than a predefined threshold. Our MDP has high complexity and dimensions regarding the state space, action space, and transition probabilities; therefore, we can not provide the optimal decision rules (exact solutions) for SAIRPs of increasing size. Thus, we propose high-quality approximate solutions, heuristic and reinforcement learning (RL) methods, for stochastic SAIRPs that provide near-optimal policies for the stations.

In Chapter 3, we explore the structure and theoretical findings related to the optimal solution of SAIRP. Notably, we prove the monotonicity properties to develop fast and intelligent algorithms to provide approximate solutions and overcome the curses of dimensionality. We show the existence of monotone optimal decision rules when there is an upper bound on the number of batteries replaced in each period. We demonstrate the monotone structure for the MDP value function when considering the first, second, and both dimensions of the state. We utilize data analytics and regression techniques to provide an intelligent initialization for our monotone approximate dynamic programming (ADP) algorithm. Finally, we provide insights from solving realistic-sized SAIRPs.

In Chapter 4, we consider the problem of optimizing the distribution operations of a hub

using drones to deliver medical supplies to different geographic regions. Drones are an innovative method with many benefits including low-contact delivery thereby reducing the spread of pandemic and vaccine-preventable diseases. While we focus on medical supply delivery for this work, it is applicable to drone delivery for many other applications, including food, postal items, and e-commerce delivery. In this chapter, our goal is to address drone delivery challenges by optimizing the distribution operations at a drone hub that dispatch drones to different geographic locations generating stochastic demands for medical supplies. By considering different geographic locations, we consider different classes of demand that require different flight ranges, which is directly related to the amount of charge held in a drone battery. We classify the stochastic demands based on their distance from the drone hub, use a Markov decision process to model the problem, and perform computational tests using realistic data representing a prominent drone delivery company. We solve the problem using a reinforcement learning method and show its high performance compared with the exact solution found using dynamic programming. Finally, we analyze the results and provide insights for managing the drone hub operations.

Acknowledgments

I would like to thank my advisor, Dr. Sarah Nurre Pinkley, for her guidance and support. I am extremely grateful to her for selecting me as her Ph.D. student, being flexible, challenging me, and supporting me to conduct impactful research. My special thanks go to my TA supervisor, dissertation committee member, and role model, Dr. Kelly Sullivan. I have been incredibly fortunate to work with him, who influenced my character and career in so many ways. I cannot be more grateful for his strong belief in me, fairness, support during ups and downs, keeping me enthusiastic, and giving me just the right level of freedom. I am forever grateful for his outstanding mentorship, endless personal and professional support.

I also would like to thank my other dissertation committee members, Dr. Shengfan Zhang and Dr. Mohammad Marufuzzaman, for their vital contribution and guidance to this dissertation. I want to thank all my committee members for their valuable time to support me during my job search and every opportunity they have given me to fulfill my goals.

I would like to thank the faculty and staff of the Department of Industrial Engineering at the University of Arkansas for their help and support in many ways. Specifically, I would like to thank our department head, Dr. Edward Pohl, for his trust in me, giving me the opportunity to be a successful instructor over two semesters, selecting me for IISE and INFORMS colloquiums, and for his valuable time and support during my job search.

My special thanks go to to all my friends who have supported me and shaped my journey in Arkansas. Especially, I express my sincere gratitude to Fereydoun Adbesh, Payam Parsa, Aydin Iranzad, Maryam Alimohammadi, Parham Pouldsanj, Sahar Taji, Samira Karimi, Ali Rahimpour, Ali Balapour, Ghazaleh Salehabadi, Imann and Salman Mosleh, and Maryam Amirvaghefi.

Finally, I want to say thank you from my heart to my parents, Zohreh and Gholamreza, my sister, Elaheh, and my brother-in-law, Mehdi, who are the very first support of my life. My last appreciation goes to my true friend and wife, Elham, for her immeasurable sacrifices and unfailing love in absence of my family while I pursued this final degree.

Dedication

To Elham, Maman, and Baba.

I would like to dedicate this dissertation to my family for their tremendous support and unconditional love throughout my life. I am certain that none of my success would have been achieved without the sacrifices that my parents made. Time and again they have picked me up and I hope someday I will pick them up too.

In memory of all victims of PS752 flight.

Contents

1 Introduction **1**

Bibliography **5**

2 A Stochastic Scheduling, Allocation, and Inventory Replenishment Problem for Battery Swap Stations **6**

2.1 Introduction 6

2.2 Background 11

2.3 Problem Statement 16

2.3.1 Modeling Assumptions 25

2.4 Solution Methodology 25

2.4.1 Average Capacity Approximation 26

2.4.1.1 Disaggregated Markov Decision Process Model 27

2.4.1.2 Monte Carlo Simulation 35

2.4.2 Exact Solution Method: Backward Induction 37

2.4.2.1 Effect of the Problem Size 37

2.4.3 Approximate Solution Methods 37

2.4.3.1 Heuristic Benchmark Policy 38

2.4.3.2 Reinforcement Learning Approach 43

2.5 Computational Results 46

2.5.1 Data 47

2.5.2 Latin Hypercube Designed Experiment 48

2.5.3 Performance of the Approximate Solution Methods 56

2.5.3.1 Parameter Specifics for the Approximate Methods 57

2.5.3.2 Analysis of the Experimental Results 60

2.6 Conclusions and Future Research 64

Bibliography **67**

3 A Monotone Approximate Dynamic Programming Approach for the Stochastic Scheduling, Allocation, and Inventory Replenishment Problem: Applications to Drone and Electric Vehicle Battery Swap Stations **76**

3.1 Introduction 76

3.2 Literature Review 80

3.3 Problem Statement 84

3.4 Theoretical Results 93

3.4.1 Monotone Policy 94

3.4.2 Monotone Value Function 96

3.5 Solution Methodology 97

3.5.1 Exact Solution Method: Dynamic Programming 97

3.5.2 Approximate Dynamic Programming Solution Methods 98

3.5.2.1 Monotone ADP with Regression-Based Initialization 100

3.5.2.2 Stepsize Function 103

3.6 Computational Results 104

3.6.1 Explanation of Data 105

3.6.2 Regression-Based Initialization 107

3.6.3 Latin Hypercube Designed Experiments for Modest SAIRPs 108

3.6.4 Monotone ADP Results and Performance for Realistic-sized SAIRPs . . . 118

3.7 Conclusions 122

Bibliography **125**

Appendices **132**

3.A Monotonicity of the Second Dimension of the State 132

3.B Monotonicity of Value Functions 142

3.C Algorithmic and Experimental Parameter Settings 148

4 Drones for Medical Delivery Considering Different Demands Classes: A Markov Decision Process Approach for Managing Health Centers Dispatching Medical Products 149

4.1 Introduction 149

4.2 Related Work 153

4.3 Problem Description and Formulation 156

4.4 Solution Methodology 164

4.4.1 Backward Induction 164

4.4.2 Reinforcement Learning 165

4.5 Computational Results 166

4.5.1 Data 168

4.5.2 Discussion and Analysis 172

4.5.2.1 Comparing Results of BI and RL 172

4.5.2.2 Analysis on the Number of Required Batteries 176

4.5.2.3 Demand Classification Contribution 177

4.6 Conclusion 179

Bibliography 181

5 Conclusions and Future Work 188

List of Figures

2.1 A depiction of the interaction between first- and second-level inventory for a SAIRP in the context of an Economic Order Quantity problem. 7

2.2 Percentage change in the optimal expected total reward of the aggregated vs. disaggregated MDP models. 31

2.3 Optimal policies of the aggregated versus disaggregated MDPs for Scenario 5 when realized demand equals the mean demand. 32

2.4 Optimal policies of the aggregated versus disaggregated MDPs for Scenario 35 when realized demand equals the mean demand. 33

2.5 Graphical representation of heuristic benchmark policy actions. 40

2.6 Power price fluctuation over one week. 47

2.7 Mean demand over time. 47

2.8 Action taken over time for a sample path of the optimal policy of Scenario 31 when realized demand equals the mean demand. 52

2.9 Number of full batteries and average capacity for the optimal policy over time for Scenario 31 when realized demand equals the mean demand. 52

2.10 Demand and met demand for the optimal policy for scenario 31 when realized demand equals the mean demand. 53

2.11 Number of full batteries and average capacity for the optimal policy for scenario 22 when realized demand equals the mean demand. 53

2.12 Number of full batteries and charging/discharging actions for the optimal policy over time for Scenario 35 when realized demand equals the mean demand. . . 54

2.13 Correlation between the three factors and the number of batteries replaced, the optimal expected total reward, the amount of demand satisfied, and the number of batteries charging under the optimal policy. 55

3.1 Diagram outlining the timing of events for the SAIRP model. 88

3.2 Power price fluctuations over December (realistic-sized SAIRP) and the week of Dec. 12-18 (modest Size SAIRP), 2016 in the Capital Region, New York. . . . 106

3.3 Mean demand for swaps over time. 107

3.4 An instance of the optimal values over states in two consecutive decision epochs 108

3.5 Expected total reward convergence of MADP and MADP-RB using different Stepsize functions vs optimal expected reward. 115

3.6 Sample paths of average capacity over time horizon for three types of LHS scenario when realized demand equals means demand. 121

3.7 Sample path of the policy for scenario 15 when realized demand equals mean demand. 122

3.8 Demand and met demand for scenario 15 based on the sample path when realized demand equals mean demand. 123

4.1 An example of demand classification based on the distance between the location of demand and the drone hub. 157

4.2 An instance of state transition for a single battery. 159

4.3 Locations of hospitals (demand nodes), the swap station located in Zipline drone hub, and airports in Rwanda. 170

4.4 Pattern of patients arrivals to a hospital over a day. 171

4.5 Expected total reward convergence of RL method. 174

4.6 Sample paths of states (I, IV, VI), optimal policies (II, V, VII), and demands (III, VI, IX) for $\rho^{21} = 0.5, 1, 2$ when the realized demand of either class equal mean demand. 175

4.7 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. 178

4.8 Average percentage of met demand for 500 sample paths when $\rho^{21} = 0.5$ using different models and solution methods. 179

List of Tables

2.1 Size of the state space for the aggregated and disaggregated-MDP models. 29

2.2 Example showing the size of the state space for the aggregated and disaggregated MDP models. 30

2.3 Differences in the simulation tracking each individual battery capacity as compared to the average capacity progression calculated in the SAIRP model. 36

2.4 Notation used in the reinforcement learning algorithm. 44

2.5 Factors with associated low and high values for use in the Latin hypercube designed experiment. 50

2.6 Scenarios and results of the backward induction algorithm for the Latin hypercube designed experiment. 51

2.7 Parameters used for the heuristic benchmark policy. 57

2.8 Performance comparison of the approximate solution methods. 62

3.1 Notation used in the ADP algorithms. 100

3.2 Time and memory used for different size of SAIRPs using memory intensive and compute intensive methods. 110

3.3 Factors with associated low and high values for use in the Latin hypercube designed experiment. 111

3.4 Optimality gap (%) of MADP, MADP-M, MADP-RB, and MBI. 113

3.5 Comparison between convergence over iterations using MADP and MADP-RB. . 116

3.6 Comparison between the average optimality gap of different approximate solution methods. 118

3.7 Results of realistic-sized SAIRPs for the latin hypercube designed experiment. . 119

4.1 Notation used in the reinforcement learning algorithm. 166

4.2 The data associated with blood unit delivery using Zipline drones in Rwanda, Africa. 169

4.3 The expected total reward and computational time of solving SA-MCD with $\rho^{21} = 0.5$ and $M = 15$ using BI and RL. 173

4.4 Computational time, memory used, and average percentage of met demand over time for 500 sample paths when $\rho^{21} = 0.5$ using BI and RL methods. 174

4.5 Average met demand and policies over time for 500 sample paths for different values of ρ^{21}. 176

4.6 Computational time, memory used, and average percentage of met demand over time for 500 sample paths when $\rho^{21} = 0.5$ using the RL method. 177

List of Published Papers

Chapter 2:

Asadi, A. and Nurre Pinkley, S., “A Stochastic Scheduling, Allocation, and Inventory Replenishment Problem for Battery Swap Stations,” *Transportation Research Part E: Logistics and Transportation Review*, 2021, Vol. 146, pp. 102212.

Chapter 3:

Asadi, A. and Nurre Pinkley, S., “A Monotone Approximate Dynamic Programming Approach for the Stochastic Scheduling, Allocation, and Inventory Replenishment Problem: Applications to Drone and Electric Vehicle Battery Swap Stations,” *Transportation Science* (under revision) (2021).

Chapter 4:

Asadi, A. and Nurre Pinkley, S., “Drones for Medical Delivery Considering Different Demands Classes: A Markov Decision Process Approach for Managing Health Centers Dispatching Medical Products,” *European Journal of Operational Research* (under review) (2021).

1. **Introduction**

Intelligent systems, including humans, organizations, and companies, cope with challenges to make desirable decisions to survive and flourish in our uncertain world. Notably, over the past 80 years, researchers and scientists of various perspectives and backgrounds have provided solutions for such challenges in different fields and applications. This research presents mathematical models and solution methods for making optimal/near-optimal decisions under uncertainty. Specifically, we focus on optimization problems for managing Electric vehicles (EV) and drones swap stations. EVs and drones promise to transform transportation, delivery, and supply chain systems. A swap station is an infrastructure or a physical location wherein we swap depleted batteries with recharged batteries quickly and recharge batteries to be used in anticipation of demand. Swap stations will play a vital role in the transformation as they can overcome EV and drone adoption barriers, including long recharge times, limited drive/flight range, and battery degradation.

In Chapter 2, we introduce a novel class of stochastic scheduling allocation and inventory replenishment problems (SAIRP) for managing internal operations in swap stations. In SAIRPs, we aim to determine the recharging, discharging, and replacement decisions at a swap station over time to maximize the expected total profit. The profitability of the swap stations is vital for continuous operations and absorbing investment in the stations. SAIRPs incorporate the interaction between two levels of inventory, battery charge and battery capacity, where recharging and discharging a battery lead to battery capacity degradation, and the level of battery capacity restricts the amount of stored charge inside a battery. This integration is crucial as it relaxes the assumption that recharging/discharging batteries do not deteriorate their quality (capacity). Besides, it ensures customers receive high-quality batteries as we prevented recharging/discharging and swapping when the average capacity of batteries is lower than a predefined threshold.

We model the problem as a finite horizon Markov Decision Process (MDP) model to capture the non-stationary elements of battery swap stations over time, including mean battery swap demand, recharging price, and discharging revenue. The state of the system is denoted by the average capacity and the number of fully-charged batteries. To verify our choice of average capacity (aggregating the state space) over tracking individual battery capacities, we use a disaggregated MDP model and a Monte Carlo Simulation and show the aggregation's benefit in reducing the problem size and complexity while not sacrificing the quality of the solutions. Despite the aggregation, we show that our MDP still has high complexity and dimensions regarding the state space, action space, and transition probabilities. Therefore, we can not provide the optimal decision rules (exact solutions) for SAIRPs of increasing size.

Thus, we propose two high-quality approximate solutions for stochastic SAIRPs that provide near-optimal policies for the stations. The first approximate solution method is a heuristic benchmark policy that dynamically selects actions according to the values for input parameters and the present decision epoch, which are discrete times in which we make decisions. The benchmark policy is a constructive heuristic algorithm that creates a solution dictating the action to take when in each state and time. We empirically tested many rules that indicate when and how many batteries are charged and replaced over time to design the heuristic benchmark policy. We also leverage the heuristic policy to intelligently initialize the second approximate solution method, a double pass with the heuristic policy initialization (DHPI) reinforcement learning (RL) method. After intelligent initialization, the DHPI RL method determines the actions and updates the states when moving forward in time and updates the value of the visited states stepping backward in time over iterations before a stopping criterion is satisfied. As a result, we show that the DPHI RL method can solve large-scale problems and overcome the curses of dimensionality.

In Chapter 3, we focus on the theoretical findings regarding the monotonicity of the optimal policy and value functions of SAIPRs that is motivated by exploiting efficient algorithms that require less computational effort to find optimal policies and increase the ability to solve larger problem instances (Puterman, 2005). In this chapter, we prove that the stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state. Then, we demonstrate the existence of a monotone non-increasing optimal policy in the second dimension of the state when there is an upper bound on the number of batteries replaced in each period. Moreover, we prove that the MDP value function of the stochastic SAIRP is monotonically non-decreasing in the first, second, and both dimensions of the state. By showing these theoretical properties, we have a foundation for the development of efficient solution methods. Exploiting the monotonicity of the value function, we use a monotone approximate dynamic programming algorithm proposed by Jiang and Powell (2015) and enhance it with adding a regression-based initialization. We show the value of using monotonicity property and intelligent regression-based initialization on the performance of the solution method using sets of designed experiments for modest and realistic-sized SIARPs instances. Finally, we provide an extensive analysis and derive insights from solving realistic-sized SAIRPs for managing operations in a swap station.

In Chapter 4, we study medical supplies delivery using drones in the presence of uncertainty. Drone delivery is an innovative, low-contact way to distribute essential medical supplies such as blood units and vaccines to communities located in congested or remote areas where roads and other transportation means are not viable options (Dhote and Limbourg, 2020). In our setting, a swap station is located in a drone hub or dispatching center. Our goal is to optimize a medical drone delivery system's operation in the swap stations that consider drone delivery challenges, including limited flight range, long recharging times, and the need to dispatch drones to different geographic locations generating stochastic demands for medical supplies. We classify the demand based on the distance between the swap station and demand locations. We link the level of charge inside batteries to demand classes such that the demand of each class can be satisfied with batteries having the same or higher level of charge. We propose a stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD) model to find optimal recharging actions that maximize the expected total weighted met demand. We use the Markov decision process (MDP) to model the stochastic SA-MCD. It incorporates a multi-dimensional state and action space and a complex, large transitions probability function, making SA-MSC suffer from the curses of dimensionality as the problem size increases. We propose an RL method with an exploration feature (ε-greedy policy) 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 (increase the exploitation rate) to make the algorithm converge as it proceeds toward iterations.

In this chapter, we incorporate a case study influenced by Zipline, a drone delivery company that delivers blood and other medical supplies from a hub in Muhanga district, Rwanda. We 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 SA-MCDs to manage the internal distribution operations of the swap station using different sets of computational experiments. We show the high performance of our RL method in terms of the optimality gap and average percentage of the met demand when an exact solution is available for comparison. We observe that 15 drones deployed by Zipline are not sufficient to satisfy 100% of the stochastic demand that necessitates solving larger problem instances using our RL method. We explain the relationship between the number of drones in the station and the amount of met demand using the exact and approximate solution methods. We show the contribution of demand classification and using our RL solution method, which outperforms the exact solutions derived from the model with no demand classification. We provide extensive analysis on the impact of changing the parameter controlled by the drone delivery company, which is the incentive of demand satisfaction from lower-class demand using higher-level charged batteries.

Bibliography

Dhote, J. and Limbourg, S. (2020). Designing unmanned aerial vehicle networks for biological material transportation – The case of Brussels. *Computers and Industrial Engineering*, 148:106652.

Jiang, D. R. and Powell, W. B. (2015). An approximate dynamic programming algorithm for monotone value functions. *Operations Research*, 63(6):1489–1511.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Hoboken, New Jersey, 1st edition.

2. A Stochastic Scheduling, Allocation, and Inventory Replenishment Problem for Battery Swap Stations

Amin Asadi Sarah Nurre Pinkley

2.1 Introduction

Optimizing the operations at battery swap stations for electric vehicles (EVs) and drones enable their pervasive adoption in many industries including transportation (Mutzabaugh, 2017), delivery (Weise, 2017; DHL Press Release, 2016), agriculture (Jensen, 2019), and disaster response (Soergel, 2016). A battery swap station is a physical entity which enables the (autonomous) exchange of a EV or drone's depleted battery for a full-charged battery (Widrick et al., 2018). This exchange occurs within a few seconds which is a significant improvement over the at least hour-long recharging wait time necessary at charging stations (Tesla, 2017). To enable the quick exchange, battery swap stations need to recharge previously swapped depleted batteries resulting in fully-charged batteries available for future swap demand. However, we must consider the fundamental interaction where the recharging and use of battery charge is the direct cause of battery capacity degradation. Battery capacity limits the amount of battery charge held and battery charge dictates the flying/driving time and distance. As the capacity degrades, the swap stations must determine when to replace batteries to ensure a minimum service and performance level for customers. Thus, optimizing the operations of a battery swap station is complex and must consider this fundamental interaction when making recharging and replacement decisions over time.

Although operating battery swap stations is complex, they reduce many barriers inhibiting EV and drone adoption. One specific barrier that is reduced is the wait time necessary for conducting battery recharging at charging stations. The recharging done at charging stations is often fast charging which is beneficial for reducing customer wait time but is harmful as it causes battery degradation (Rezvani et al., 2015; Saxena et al., 2015; Lacey et al., 2013; Shirk and Wishart, 2015). Battery degradation reduces battery capacity and in turn, the maximum amount of battery

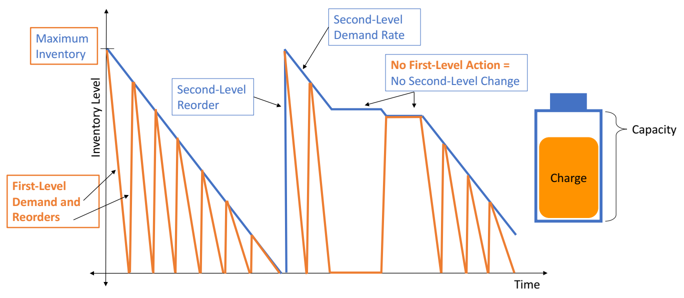


Figure 2.1: A depiction of the interaction between first- and second-level inventory for a SAIRP in the context of an Economic Order Quantity problem.

charge stored. As swap stations are recharging batteries in anticipation of future demand, they can use regular rate charging which causes less battery degradation. Less battery degradation results in fewer necessary costly battery replacements and a reduction in the environmental burden from discarded batteries (Gardiner, 2017). Due to the centralized inventory of batteries at a swap station, batteries can be discharged back to the grid using battery-to-grid (B2G) or vehicle-to-grid (V2G) (Sioshansi and Denholm, 2010). B2G and V2G enable the discharged battery energy to act as a supply point within a smart grid. When done at strategic times, the discharged energy can aid with load balancing initiatives (Peng et al., 2012; Göransson et al., 2010; Wang et al., 2011) and reducing the fluctuations from variable renewable energy sources (Dunn et al., 2011).

We consider the problem of determining the optimal charging, discharging, and replacement actions at a swap station when explicitly considering battery degradation and uncertain swap demand. To solve this problem, we introduce a novel stochastic scheduling, allocation, and inventory replenishment problem (SAIRP). A stochastic SAIRP captures the interactions between the states and actions of first-level battery charge inventory and second-level battery capacity inventory. For first-level inventory, the swap station must determine when batteries are charged and discharged. For second-level inventory, the swap station must determine when a battery is discarded from inventory and replaced with a new full-charge and full-capacity battery. The key interaction in SAIRPs is that the first-level charge and discharge actions cause the depletion of second-level capacity inventory which leads to the need for battery replacement.

In Figure 2.1, we present a simplified SAIRP for one battery in the context of a traditional

Economic Order Quantity (EOQ) model (Greene, 1997). This is a generic example in hopes to demonstrate the mechanics of the problem of study. The orange lines represent first-level battery charge inventory and the blue lines represent second-level battery capacity inventory. Focusing on the orange lines, after each use or discharge (downward slope) the battery can be recharged (upward slope) up to the capacity (blue line). The fundamental interaction between the two integrated inventory levels is that battery capacity is depleted only when battery charge is used (either in an EV or drone or through discharging) or recharged. When a battery is deemed to not have sufficient capacity, it must be replaced. The stochastic SAIRPs we consider for swap stations are significantly more complex than this simplified SAIRP. This complexity stems from the need to schedule and allocate multiple batteries at the swap station simultaneously, our consideration of an uncertain demand rate for swaps, and our inclusion of realistic varying charging costs over time.

We propose a Markov Decision Process (MDP) to model the stochastic SAIRP. An MDP is a Markov chain with reward associated with transitions and actions. Hence, to find the optimal solution of an MDP, we must search for the policy that optimizes the expected total reward (Puterman, 2005; Powell, 2011). For a stochastic SAIRP, a policy is a set of decision rules dictating the optimal charging, discharging, and replacement actions when in a given state and time. Although a battery swap station will be continuously operating over time, an infinite horizon MDP is not a suitable modeling approach for a problem with time-varying, nonstationary components. Thus, we use a finite horizon MDP due to the nonstationary elements for this problem including the time-varying mean arrival of swap demand and the time-varying charging costs. In the MDP model, we represent the state using two dimensions where the first dimension represents the number of fully charged batteries at the swap station and the second dimension represents average capacity of all batteries at the swap station. In this design, we assume that a battery is either fully charged or depleted. Furthermore, to reduce the curses of dimensionality, we model the aggregate average capacity of batteries at the swap station rather than each battery individually. Using both a disaggregated MDP which models each battery's individual capacity and a Monte Carlo simulation, we show that aggregating capacity has significant advantages in terms of computational time without sacrificing precision. We represent the action space using two dimensions where the first dimension represents the number of batteries selected for charging (positive value) or discharging (negative value) and the second dimension represents the number of batteries replaced.

For each discrete time, denoted a decision epoch, we seek to determine the optimal action for each state that maximizes the expected total reward. We calculate the expected total reward as the expected swap station profit equal to revenue minus cost. The swap station earns revenue from capacity-dependent swap revenue and discharging batteries back to the grid, and incurs costs from charging and replacing batteries. We use backward induction to identify the optimal policy comprised of decisions rules which indicate the optimal actions to take when each state and time. As our problem suffers from the curses of dimensionality, backward induction is not an effective method for determining optimal policies for realistic-sized problem instances. As is common with the broader stochastic optimization community, we use approximate solution methods to solve our complex problem. First, we propose a heuristic benchmark policy, similar to a constructive heuristic (Schneider and Kirkpatrick, 2006), which is a rule/algorithm used to generate a policy in a matter of seconds. We first propose the general outline of the benchmark policy and then discuss how specific parameter values were used for our computational experiments. Second, we propose a reinforcement learning approach with heuristic benchmark policy initialization. Reinforcement learning (RL) is a common approach to solve the large-scale problems suffering from the curses of dimensionality (Powell, 2011; Sutton and Barto, 2018). Our RL method benefits from smart initialization through feeding the value and policies generated by evaluating the heuristic benchmark method for simulated sample paths. Furthermore, in our RL method, we use a double pass updating procedure to update the visited states in the forward pass and update the values in the backward pass (see (Powell, 2011) for more information on double pass).

Using the MDP model of the stochastic SAIRP, we perform extensive computational experiments. We perform a Latin hypercube designed experiment to identify the optimal policies for a swap station with different characteristics. We solve each instance of the designed experiment using our three solution approaches: backward induction (BI), our heuristic benchmark policy, and our reinforcement learning approach with heuristic benchmark policy initialization. For each of the instances, we analyze the results to deduce insights about the optimal charging, discharging, and replacement policies and the performance of the different solution methods. Experimentally, we show that the solutions generated by the benchmark policy are near-optimal with regards to the expected total reward and amount of demand met. Moreover, the proposed reinforcement learning result indicates even better performance in terms of the optimality gap. It can also project the value and policy for large-sized instances of the problem.

The **main contributions** of this research are as follows: (i) we are the first to introduce a stochastic scheduling, allocation, and inventory replenishment problem (SAIRP) that links the actions for first-level (battery charge) inventory to the level and needed actions for second-level (battery capacity) inventory; (ii) we develop a two-dimensional MDP model which determines the optimal first-level inventory charge and discharge actions and second-level capacity replacement actions at a swap station that faces multiple stochastic, nonstationary swap demand levels, nonstationary charging costs, nonstationary discharging revenues, and capacity dependent swap revenue; (iii) we validate the modeling choice of using average capacity as opposed to each individual battery capacity using a disaggregated MDP and Monte Carlo simulation; (iv) we propose a competitive heuristic benchmark that overcomes the curses of dimensionality; (v) we propose a reinforcement learning approach with heuristic benchmark policy initialization capable of generating high-performance solutions for different sizes of SAIRPs; and (vi) we deduce insights from the results of a Latin hypercube designed experiment which uses backward induction, the benchmark policy, and the reinforcement learning approach to solve many instances of a stochastic SAIRP.

The remainder of this chapter is organized as follows. In Section 2.2, we summarize relevant literature on the use of optimization approaches for EV and drone applications. In Section 2.3, we present a two-dimensional Markov Decision Process to model a stochastic scheduling, allocation, and inventory replenishment problem. In Section 2.4, we validate our modeling assumptions and outline our solution methodology. We present the results of a designed experiment in Section 2.5 which we use to deduce insights about swap station management. We conclude this chapter with a summary of the contributions and outline future research directions in Section 2.6.

2.2 Background

Swap stations have many benefits including reducing the wait time necessary for customers, reducing battery degradation (equivalently battery waste), and enabling B2G or V2G. With the growth in EVs, drones, and other battery operated devices, there is an increased interest in the potential of battery swap stations. Swap stations are not the only viable option for overcoming barriers to EV and drone adoption. There is an extensive body of literature that examines the use of battery charging stations (e.g., (Liu, 2017; Chen et al., 2018; Schroeder and Traber, 2012; Sathaye and Kelley, 2013)). We proceed by discussing relevant literature for the application areas of EV and drone swap stations with particular focus on works using optimization techniques.

Researchers have examined EV and drone battery swap stations from many different perspectives. First, we review the research most similar to this work which examines optimizing the internal operations of an EV or drone battery swap station faced with uncertainty. Widrick et al. (2018) propose an MDP model to determine the optimal number of batteries to charge and discharge over time when considering the uncertain arrival of battery swaps. Although, we consider similar charging and discharging decisions, our work is distinctly different from Widrick et al. (2018) as we explicitly consider battery capacity degradation. The consideration of battery capacity adds significant complexity and necessitates a two-dimensional MDP to capture the necessary state and actions with regard to capacity. Worley and Klabjan (2011) also consider the internal operations when faced with uncertain demand but consider the optimal purchasing and charging over time to minimize total cost. The decision regarding purchasing batteries has similarities to battery replacement; however, we clarify that SAIRPs consider a static number of batteries at the swap station, do not explicitly consider the decision regarding how many batteries to purchase prior to opening the swap station, and instead consider the timing and amount of batteries replaced while the swap station is operating. Schneider et al. (2018) consider short-term charging decisions at a swap station coupled with long-term decisions regarding the number of batteries to purchase and the number of charging bays to consider at the swap station. Schneider et al. (2018) do use the term *battery capacity*, however they consider capacity as the number of charging bays (i.e., number of batteries that can be charged at one time) which is distinctly different from SAIRP capacity representing the maximum amount of charge a battery can hold. To the best of our knowledge, we are the first to consider optimizing the internal operations of a battery swap station when faced with uncertain swap demand and battery degradation.

Other researchers seek to optimize the internal operations of an EV or drone battery swap station but ignore uncertainty. Nurre et al. (2014) use a mixed integer programming formulation to determine the number of batteries to charge, discharge, and swap over time. They propose load balancing policies to encourage charging and discharging at desirable times. Recently, researchers have examined the idea of battery degradation in a deterministic setting. Kwizera and Nurre (2018) consider a deterministic two-level integrated inventory problem for a drone battery swap station where battery degradation limits the maximum drone fly time which in turn limits the jobs able to be completed from the swap station without replacement. Similarly, Park et al. (2017) propose a deterministic model to determine the battery charging schedule and assignment of jobs to minimize battery degradation. Rather than considering the entire set of decisions at once, they solve two subproblems, one which determines the battery charging schedule and another which determines the assignment of batteries to jobs.

There are other studies which examine swap stations without focusing on the internal operations. For example, Adler et al. (2016) developed an integer programming model to determine the optimal locations for swap stations. Shavarani et al. (2018) use a facility location model to determine the number and locations of swap stations for a drone delivery problem. When considering the interaction between the power grid and swap stations, Pan et al. (2010) propose a two-stage stochastic model to locate swap stations. Likewise, Gao et al. (2012) measure the effectiveness of a battery swap station for increasing the reliability of the power system working with wind power. Others examine routing drones through one or many stations to minimize total travel distance (Kim et al., 2013; Kim and Moon, 2019).

In parallel to the examination of swap stations among researchers, several companies have implemented swap stations or are developing technology to enable drone or EV battery swap. For example, the Chinese automobile manufacturer, NIO, has 18 operational swap stations and aims to increase the number of swap stations to 1100 by 2020 (Lambert, 2018). Furthermore, companies with a fleet of vehicles, such as BJEV, found the idea of swap stations viable. By 2017, BJEV established more than 100 swap stations for its electric cabs and aims to expand to over 100 cities in China. They plan to build 3000 swapping stations by 2020 (Fusheng, 2019). Additionally, there are many companies using drone swap stations and examining the physical mechanics of a drone swap station (Markoff, 2016; AIROBOTICS, 2020; McNabb, 2017; Popper, 2016).

Our work can be interpreted as discrete-time sequential decision-making for managing inventory and replacing equipment in the presence of uncertainty in the system. There is a rich literature of using MDP models for inventory management problems under uncertainty. For example, Rong (2012) solves an MDP model with dynamic programming to derive the optimum order point and quantity for a single product, multi-stage inventory system. Cheng and Sethi (1999) propose an MDP model for an inventory management problem where consumer demand can be affected by the promotions offered by the company. Ahiska et al. (2013) utilize an MDP model to find the structure of optimal policies for purchase and inventory control problems in a stochastic environment. Moreover, MDPs have been widely used to model equipment replacement problems. Venkatesan (1984) provides the foundation for a single-product, single-equipment production-inventory MDP system. It determines the optimal policies for static and dynamic system conditions, where it is assumed that products can be replaced instantaneously. Nair (1995) finds the optimal decision concerning investment in new equipment/technology wherein several sequential technologies may appear with different associated costs and revenue in each decision epoch. Chan and Asgarpoor (2006) look for the optimum maintenance policy for a component where both random failure and failures due to deterioration exist. Abeygunawardane et al. (2013) use an MDP to optimize a maintenance program considering product aging. Besides inventory management and equipment replacement, MDPs are broadly used for modeling problems dealing with electric vehicle and drone applications. For instance, Iversen et al. (2014) propose an MDP based algorithm to solve the problem of finding the optimal charging policies of EV batteries when incorporating uncertainty in human driving behaviors. Al-Sabban et al. (2013) used an MDP to find the optimal flight path between a predefined origin and destination for drones empowered by wind energy that is uncertain in terms of magnitude and direction. Similarly, Baek et al. (2013) aim to find the optimal grid-based movement policy for drones. Fu et al. (2015) used an MDP model for managing the navigation of drones on 2-D planes such that no collisions happen in the drone flight route. Widrick et al. (2018) developed an inventory control MDP for a swap station that only considers battery charge.

Our work can be viewed as a dynamic inventory and allocation problem. This problem has been well studied, including fundamental research (Scarf, 1960; Karlin, 1960; Peterson and Silver, 1979) but also research examining how to approximate the optimal policy. There are different approaches to approximate optimal policies including myopic policies, look-ahead policies (known as rolling horizon procedure, receding horizon procedure, and model predictive control in different communities), policy function approximations, and value function approximations (Powell, 2011). We proceed by presenting examples of optimal policy approximation for dynamic inventory management problems. Federgruen and Zipkin (1984) approximate a dynamic ordering allocation problem with stochastic demand, lead time, and backorder by reducing the multiple-location inventory problem to a single-location inventory problem. Somarin et al. (2017) formulate a dynamic inventory allocation problem as an MDP and propose a heuristic method to find near-optimal policies using value function approximations. Fu et al. (2019) used a model predictive control (look-ahead policy) to find optimal solutions in a supply chain wherein the suppliers are interacting to meet demands. Similarly, Hai et al. (2011) apply model predictive control to optimize the cost of handling dynamic inventory management in a supply chain context. Tang et al. (2015) aim to find a look-ahead control policy for a conveyor-serviced production station wherein parts and demands randomly arrive in the system. For further study, we refer the reader to the works of (Bookbinder and H'ng, 1986; Anupindi et al., 1996; Cheevaprawatdomrong and Smith, 2004) to see the application of the rolling horizon method to handle dynamic inventory problems.

To solve large-scale inventory and allocation problems, researchers use various approaches of reinforcement learning (RL) and approximate dynamic programming (ADP), including a one-step temporal difference (TD) (Roy et al., 1997), case-based myopic RL (Jiang and Sheng, 2009), dual heuristic programming (Shervais et al., 2003), Q-Learning (Chaharsooghi et al., 2008), SARSA and simulation-based ADPs (Katanyukul et al., 2011). In our RL method, we use a look-up table for the value function approximation which is a common approach (see (Jiang and Sheng, 2009; Kwon et al., 2008)). We use a double pass procedure, forward pass in time to determine the actions and update the states and backward pass in time to update the value of the visited states. The backward pass idea is similar to the backpropagation method for training multi-layer Artificial Neural Networks (ANN). We refer the reader to (Williams, 1988; Paul Werbos, 1989; Gullapalli, 1990) for further study about backpropagation in RL and ANN.

We are the first to consider the impact of battery degradation when operating a battery swap station under uncertainty. Excluding battery capacity does lead to a simpler model; however, it fails to include an important realistic aspect that influences decision making at a battery swap station. One common way to measure battery degradation, or alternatively battery capacity, is through cycle counts. One cycle equates to one use/discharge and one charge. Thus, a battery life span can be estimated using the number of cycles. We acknowledge that there are many factors which influence battery degradation, such as temperature and depth of discharge (Plett, 2011; Abe et al., 2012; Ribbernick et al., 2015; Dubarry et al., 2011). We assume that a battery sitting in inventory experiences no degradation. Further, we use a linear degradation factor in accordance with previous research (Lacey et al., 2013; Ribbernick et al., 2015; Wood et al., 2011). We proceed by providing the MDP model for stochastic SAIRPs which explicitly considers battery degradation.

2.3 Problem Statement

In this section, we model the scheduling, allocation, and inventory replenishment problem (SAIRP) as a Markov Decision Process (MDP) considering time-varying costs from charging batteries, costs from replacing capacity-depleted batteries, time-varying revenue from discharging to the power grid, and capacity-dependent revenue from swapping batteries to meet stochastic demand over time. Importantly, the SAIRP MDP incorporates the interactions between first-level battery charge inventory and second-level battery capacity inventory. The problem experiences uncertainty from the uncertain arrival of swap demand over time which we model based on a known distribution and time-varying mean. Due to the dynamic and time-varying characteristics of the problem, we present the following finite horizon MDP which seeks to maximize the expected total reward when determining the optimal number of batteries to charge, discharge, and replace over time. In Section 2.4, we outline how we find optimal policies using backward induction and near-optimal policies using a heuristic benchmark policy for our MDP.

Decision Epochs: The set $T = \{1, \dots, N-1\}$, $N < \infty$ represents the discrete points in time when decisions are made.

States: The state of system is dynamic and varies over time. The state has two dimensions, $s_t = (s_t^1, s_t^2) \in S = (S^1 \times S^2)$, for all decision epochs $t \in T$. The first dimension, $s_t^1 \in S^1 = \{0, 1, \dots, M\}$, indicates the total number of full batteries at decision epoch t, where M denotes the total number of batteries at the swap station. The second dimension, $s_t^2 \in S^2 = \{0, \varepsilon, 2\varepsilon, \dots, 1\}$, indicates the average capacity of all batteries at the swap station represented as a percentage rounded to the nearest ε. In our model, we set a threshold, θ, which equals the lowest acceptable average battery capacity so as to ensure customer satisfaction. We do not allow batteries with an average capacity below θ to be replaced, charged, discharged, or swapped, as we assume the swap station is not operational when the average capacity drops below θ. If the average capacity is below θ, we model the average capacity as equal to 0 thereby representing all average capacities below θ. As no actions can be taken once the average capacity is in state 0, we note that 0 is an absorbing state. Instead, if the swap stations wants to maintain operations, then it must take the appropriate actions *prior* to having the capacity drop below θ to ensure customers are always guaranteed a battery with sufficient capacity (i.e., range, flight time). Contrarily, we could model the situation where swap stations may only replace batteries if the average capacity is below θ. However, this would require an expanded state space which includes all possible average capacity values just below θ that the system could feasibly transition to. Further, such a design could be equivalently represented by the presented SAIRP MDP model with the appropriately selected θ value representing end of battery life or lowest acceptable battery capacity level. As follows, S^2 is more precisely represented as $S^2 = \{0, \theta, \theta + \varepsilon, \theta + 2\varepsilon, \ldots, 1\}$.

It is important to include battery capacity to ensure the model takes a step towards better representing reality in order to best inform battery swap station operations. However, including each individual battery capacity into a model (i.e., disaggregate modeling) is computationally insurmountable for solution methods. We will demonstrate in Section 2.4.1 that using the average capacity (i.e., aggregate modeling) does not significantly shift the optimal policy as compared to the small case of the disaggregated model that is solvable. We also show that the optimal expected total reward does not dramatically change between the disaggregated and aggregated model for the majority of cases.

Actions: The action of the system is dynamically selected such that it optimizes the current and expected contributions. We define a two-dimensional action for each decision epoch t, $a_t = (a_t^1, a_t^2) \in (A^1 \times A^2)$. Because the actions are not fixed a priori, we must select the action a_t for each s_t from the action space. The first dimension of the action, $a_t^1 \in A_t^1 = \{\max(-s_t^1, -\Phi), \ldots, 0, \ldots, \min(M - s_t^1 - a_t^2, \Phi)\}$ indicates the total number of batteries to charge or discharge during epoch t. We define Φ to represent the number of plug-ins available for charging and discharging at the station. When a_t^1 is negative it indicates to discharge batteries and when a_t^1

is positive it indicates to charge. We can discharge up to the minimum of the number of plug-ins and the number of full batteries. We can charge up to the minimum of the number of plug-ins and the number of depleted batteries. For clarity, we further define a_t^1 as follows:

$$a_t^{1+} = \begin{cases} a_t^1 & \text{if} \quad a_t^1 \geq 0, \\ 0 & \text{otherwise}, \end{cases} \tag{2.1}$$

$$a_t^{1-} = \begin{cases} |a_t^1| & \text{if} \quad a_t^1 < 0, \\ 0 & \text{otherwise}. \end{cases} \tag{2.2}$$

The second dimension of the action, $a_t^2 \in A_t^2 = \{0, \ldots, M - s_t^1\}$, is the total number of batteries replaced. We assume that only depleted batteries can be replaced during epoch t and enter the system at $t+1$ fully charged. We assume that during each epoch, simultaneous charging and discharging will not occur because it can be equivalently represented as solely charging or discharging. Additionally, we assume that all plug-ins are capable of both charging and discharging and it takes one decision epoch to make a depleted battery full or vice versa. We sequence the observations and actions during an epoch in the following order. At the start of epoch t, the state of the system $s_t = (s_t^1, s_t^2)$ is known. First, a_t^2 batteries are selected to be replaced, followed by choosing a_t^1 batteries to charge/discharge. Then, the $s_t^1 - a_t^{1-}$ full batteries remaining in inventory are used to satisfy demand. At the conclusion of epoch t, the replaced batteries arrive fully charged with full capacity, the charged batteries become fully charged, the discharged batteries become depleted, and the swapped batteries become depleted. The set of feasible actions depend on the current state at time t, s_t, thus, we define A_{s_t} to indicate the feasible set of actions that can be taken when in state s_t at time t. With this design, we set assumptions describing how and when an action may be selected, including the timing of the charging/discharging and replacement between two consecutive decision epochs. However, these assumptions do not prescribe which action should be taken when in each state and time. Instead, we seek to find optimal policies which prescribe how many batteries to charge/discharge and replace for every s_t. Therefore, our model should not be mistaken with the Markov reward process in which the actions are fixed.

Transition probabilities: The state of the system transitions based on probabilities due to the uncertain arrival of battery exchanges. Thus, the number of full batteries at the swap station and the average battery capacity dynamically changes based on the combination of the realized demand and the optimal action. We define D_t as the random variable representing demand for battery exchanges in period t; thus, the met demand in time t equals $\min\{D_t, s_t^1 - a_t^{1-}\}$ where a_t^{1-} is the number of batteries which are not available due to discharging. The first dimension of the state of the system, or equivalently the number of full batteries, dynamically transitions over time according to Equation (2.3).

$$s_{t+1}^1 = s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - \min\{D_t, s_t^1 - a_t^{1-}\}. \tag{2.3}$$

We note, this first dimension transition occurs only if $s_t^2 \geq \theta$ which forces the swap station to only operate if they ensure batteries have sufficient capacity (equivalently a service level). To calculate the transition of the second dimension of the state, representing the average capacity, we define δ^C to equal the degradation in capacity for one cycle which we fully attribute to the charging and discharging actions. We define function, $g^2(s_t^1, s_t^2, a_t^1, a_t^2)$, to represent the transition from one decision epoch to the next for the second dimension of the system in Equation (2.4).

$$g^2(s_t^1, s_t^2, a_t^1, a_t^2) = s_{t+1}^2 = round\left(\frac{1}{M}\left[(s_t^2 - \delta^C)(a_t^{1+} + a_t^{1-}) + a_t^2 + s_t^2(M - a_t^{1+} - a_t^{1-} - a_t^2)\right]\right). \tag{2.4}$$

Equation (2.4) consists of three terms divided by the total number of batteries thereby dynamically calculating the change in average capacity over time. The first term captures the change in average capacity when batteries are both charged and discharged. The second term adds 1 multiplied by the number of batteries replaced arriving at full capacity. Lastly, the third term captures the batteries which are not charged, discharged, or replaced and remain in inventory at the same capacity value. This supports our assumption that all batteries swapped at time

t have a capacity equal to the average capacity of batteries at the swap station. In this design, we assume that customers use the swap station frequently enough such that the average capacities of batteries internal and external to the swap station are similar. We discretize the possible average capacities in increments of ε and thus, our use of $round()$ indicates a traditional rounding function to the nearest ε.

In our model, we define absorbing state $0 \in S^2$ for batteries with average capacity below θ. To ensure the absorbing property, we define the only feasible action for $s_t = (s_t^1, 0)$, for any s_t^1 as $a_t = (0,0)$ thereby preventing charging, discharging, replacement, swapping, and transition out of this state. In this design, the swap station can no longer operate if the average battery capacity is below θ. Thus, if the swap station wants to continue operating it would need to replace batteries before allowing the average battery capacity to drop below θ. As a result, the transition for the second dimension of the state space is precisely presented in accordance with Equation (2.5).

$$f^2(s_t^1, s_t^2, a_t^1, a_t^2) \quad = \quad s_{t+1}^2 \quad = \quad \begin{cases} g^2(s_t^1, s_t^2, a_t^1, a_t^2) & \text{if} \quad g^2(s_t^1, s_t^2, a_t^1, a_t^2) \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

We denote the probability of transitioning to state $j = (j^1, j^2)$ at time $t+1$ when at time t the system is in state $s_t = (s_t^1, s_t^2)$ when action $a_t = (a_t^1, a_t^2)$ is taken as follows:

$$p(j|s_t,a_t) = \begin{cases} p_{s_t^1+a_t^2+a_t^{1+}-a_t^{1-}-j^1} & \text{if } j^2 = f^2(s_t^1,s_t^2,a_t^1,a_t^2), \\ & \text{and } a_t^2+a_t^{1+} < j^1 \leq \\ & s_t^1+a_t^2+a_t^{1+}-a_t^{1-}, \\ q_{s_t^1+a_t^2+a_t^{1+}-a_t^{1-}-j^1} & \text{if } j^2 = f^2(s_t^1,s_t^2,a_t^1,a_t^2), \\ & \text{and } j^1 = a_t^2+a_t^{1+}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.6}$$

where $p_j = P(D_t = j)$ and $q_u = \sum_{j=u}^{\infty} p_j = P(D_t \geq u)$. As defined in Equation (2.3), the number of batteries swapped in epoch t equals $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1$, where j^1 equates to s_{t+1}^1. If no batteries are swapped in epoch t, the number of fully charged batteries at the conclusion of epoch t and start of epoch $t+1$ equals $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-}$. Alternatively, if all available fully-charged batteries are swapped in epoch t, the number of fully charged batteries at the conclusion of epoch t and start of epoch $t+1$ equals $a_t^2 + a_t^{1+}$ due to the batteries charged and replaced in epoch t.

The transition probability equals $P(D_t = s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1)$ when the demand for swaps is less than the number of batteries available for swapping and the average capacity satisfies Equation (2.5). Alternatively, if the demand for swaps is greater than or equal to the current inventory and Equation (2.5) is satisfied, the transition probability equals $P(D_t \geq s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1)$. The transition probability equals zero if j^1 is less than the number of batteries that arrive fully charged at the end of epoch t, $a_t^2 + a_t^{1+}$, j^1 is greater than the maximum number of fully charged batteries, $s_t^1 + a_t^2 + a_t^{1+} - a^{1-}$, or j^2 does not satisfy Equation (2.5).

Reward: We seek to maximize the expected total reward. We calculate the immediate reward, $r_t(s_t, a_t, s_{t+1})$, for epoch t when the system is in state $s_t = (s_t^1, s_t^2)$ and action $a_t = (a_t^1, a_t^2)$ is taken and the state transitions to $s_{t+1} = (s_{t+1}^1, s_{t+1}^2)$ depending on the realized uncertainty of the system. We can dynamically calculate the profit earned using Equation (2.7). Specifically,

$$r_t(s_t, a_t, s_{t+1}) \;=\; \rho_{s_t^2}(s_t^1 \;+\; a_t^2 \;+\; a_t^{1+} \;-\; a_t^{1-} \;-\; s_{t+1}^1) \;-\; K_t a_t^{1+} \;+\; J_t a_t^{1-} \;-\; L_t a_t^2, \quad (2.7)$$

where $\min\{D_t, s_t^1 - a_t^{1-}\} = s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - s_{t+1}^1$ equals the number of batteries swapped, K_t is the cost per battery charged during epoch t, J_t is the revenue earned per battery discharged during t, L_t is the cost per battery replaced during epoch t, and $\rho_{s_t^2}$ is the capacity-dependent revenue per battery swapped. We define $\rho_{s_t^2}$ to represent the realized swap revenue that the stations earns which is at least as large as the cost to charge a battery. This definition ensures swapping batteries is profitable which is necessary for continuously operating a swap station. Further, we define $\rho_{s_t^2}$ such that it is larger when batteries with higher average capacity are swapped and lower when the average capacity is near θ. Specifically,

$$\rho_{s_t^2} = \beta\left(1 + \frac{s_t^2 - \theta}{1-\theta}\right) = \frac{\beta(1 + s_t^2 - 2\theta)}{1-\theta}, \quad (2.8)$$

where β is set to ensure the swap revenue is never lower than the cost to recharge a battery. This is analogous to having a manufacturer of a product ensure that the revenue earned from selling the product is at least equal to the cost of production. In this context, β should be set to cover the cost of production (battery recharging) plus the profit margin per product (battery swapped). Otherwise, if β is less than the cost of production, the manufacturer (swap station) is not profitable or sustainable. We also set the swap revenue to be capacity dependent. Specifically, when the average capacity of batteries in the swap station is at the lowest operational level, i.e., $s_t^2 = \theta$, then $\rho_{s_t^2} = \beta$ or just higher than the cost to charge a battery. When the average capacity equals 1, then $\rho_{s_t^2} = 2\beta$ wherein this revenue captures a higher level of customer satisfaction due to swapping higher quality batteries. Ultimately, the swap revenue is in $[\beta, 2\beta]$ depending on how the average battery capacity, s_t^2, varies between θ and 1.

In the terminal time period only swapping occurs which means the per battery swap revenue does not change because the battery capacity only changes when recharging, discharging, or replacement occurs. Further, only batteries that are full at the start of the terminal time period can be swapped; therefore the dynamics of how charging/discharging costs change are not considered in this terminal time period. Hence, we calculate the terminal reward, $r_N(s_N)$, as the revenue from swapping fully-charged batteries provided the average capacity meets or exceeds the threshold capacity as given in Equation (2.9),

$$r_N(s_N) = \begin{cases} \rho_{s_N^2} s_N^1 & \text{if} \quad s_N^2 \geq \theta, \\ 0 & \text{otherwise}, \end{cases} \tag{2.9}$$

where the first condition is when the threshold is met or exceeded and 0 otherwise. The terminal reward is exactly the value function for the terminal time period, $u_N(s_N)$. By combining the probability transition function and the immediate reward, we are able to express the immediate expected reward function as

$$r_t(s_t, a_t) = \gamma \sum_{s_{t+1} \in S} \left[p_t(s_{t+1}|s_t, a_t) \left(\rho_{s_t^2}(s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - s_{t+1}^1) \right) \right] - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2, \tag{2.10}$$

where γ represents the discount factor. As will be described in Section 2.5, we examine short time horizons (e.g., one week) in which any changes in currency are not significant. Thus, we assume $\gamma = 1$. We denote the state and time dependent decision rules, $d_t(s_t) : s_t \to A_{s_t}$, to indicate which action $a_t \in A_{s_t}$ to select when in state s_t at decision epoch $t \in T$ with certainty. Our decision rules only depend on the current state and prescribe a single, specific action, thus, we consider deterministic Markovian decision rules (Puterman, 2005). We denote a policy π as a sequence of decision rules $(d_1^\pi(s_1), d_2^\pi(s_2), \ldots, d_{N-1}^\pi(s_{N-1}))$ for all decision epochs. The expected total reward of policy π when the system begins in state s_1 is denoted $\upsilon_N^\pi(s_1)$ and equals

$$\upsilon_N^\pi(s_1) = \mathbb{E}_{s_1}^\pi \left[\sum_{t=1}^{N-1} r_t(s_t, a_t) + r_N(s_N) \right]. \tag{2.11}$$

We seek to find a policy π^* with the maximum expected total reward. The optimal value function, $u_t^*(s_t)$, is defined as the maximum expected total reward over all policies from decision epoch t to N when in state s_t at time t. We use the backward induction algorithm, a dynamic programming technique, to determine the optimal value functions for our stochastic problem using optimality equations, or Bellman equations. With the backward induction algorithm, we use the derived optimal values to find the optimal policies. We define the optimality equations in Equation (2.12) for $t = 1, \ldots, N-1$ and $s_t \in S$.

$$u_t(s_t) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p_t(j|s_t, a_t) u_{t+1}(j) \right\}. \tag{2.12}$$

For $t = N$, we define $u_N(s_N) = r_N(s_N)$ as defined in Equation (2.9). Then moving backward in time, we let $t = N-1$ and calculate

$$u_{N-1}(s_{N-1}) = r_{N-1}(s_{N-1}, a_{N-1}) + \sum_{j \in S} p_{N-1}(j|s_{N-1}, a_{N-1}) u_N(j) \tag{2.13}$$

for all s_{N-1}. This calculation is possible because we calculate the immediate expected reward using Equation (2.10) and add this to the product of the conditional probability (from Equation (2.6)) and the value function just calculated ($u_N(s_N)$). The $u_{N-1}(s_{N-1})$ value gives us the expected value of being in state s_{N-1} from time $N-1$ to the end of the time horizon when action a_{N-1} is taken. As we do not want to implement any action and instead want to select the action that results in the highest value, we calculate Equations (2.12) and (2.14) for $t = N-1$, which determines the maximum value and associated action, respectively.

$$A^*_{s_t,t} = \text{arg max}_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p_t(j|s_t, a_t) u_{t+1}(j) \right\}. \tag{2.14}$$

Then, we move backward and let $t = N-2$ and continue this process until we reach decision epoch $t = 1$. As Backward Induction is a known optimal algorithm (Puterman, 2005), the solution to Equation (2.12) at $t = 1$ gives the optimal expected total reward for the time horizon of the problem. Thus, we emphasize that we must use optimization to determine the optimal policies which prescribe optimal actions. After determining these optimal actions, we can calculate the optimal expected total reward.

2.3.1 Modeling Assumptions

In this section, we present a summary of modeling assumptions.

1. A battery is either fully charged or depleted.

2. A battery sitting in inventory experiences no degradation.

3. Degradation occurs through charge/discharge cycles, and the battery degradation per cycle, δ^C, is linear.

4. The swap station is not operational when the average capacity drops below the threshold.

5. Only depleted batteries can be replaced during epoch t and enter the system at $t+1$ fully charged.

6. During each epoch, simultaneous charging and discharging will not occur.

7. The average capacities of batteries internal and external to the swap station are similar.

8. The station can swap all remaining fully charged batteries at the end of the time horizon if the average battery capacity is above the threshold.

2.4 Solution Methodology

In this section, we outline our solution methodology in regards to modeling assumptions and determining optimal or near-optimal policies. We model the stochastic SAIRP as an MDP and apply three solution approaches. The first solution approach is backward induction (BI), a standard solution method which is an exact algorithm guaranteeing that an optimal policy is found (Puterman, 2005). However, because the stochastic SAIRP is a large, complex problem, our model suffers from the curses of dimensionality and thus, BI is not computationally capable of finding optimal policies without running into memory errors. Thus, as is common with large problems, we heuristically solve the problem using two different solution methods. First, we propose a heuristic benchmark policy that quickly determines heuristic policies for the MDP. Next, we use a reinforcement learning approach to find near-optimal policies through an iterative process that learns and updates the present value approximations through a double pass algorithm. In the forward pass, the states are visited based on the present visited state, the realized uncertainty from samples of the stochastic demand distribution, and the taken action that maximizes the summation of the immediate reward (present contribution) and the expected contribution of the visited state in the future. The algorithm updates the visited states' rewards using the present observed value and previous approximation available for the visited states in the backward pass. All methods incorporate uncertainty and do not have perfect information. Our problem belongs to the class of stochastic problems where exogenous information, (i.e., the amount of swap demand over time for our problem) is unknown but follows a known probability distribution (Powell, 2011). First, we validate our modeling assumption where we use the average capacity of all batteries in the swap station as opposed to each individual battery capacity. Next, we describe the exact solution method, benchmark solution method, and the reinforcement learning approach for determining the expected total reward and policy.

2.4.1 Average Capacity Approximation

Due to the curses of dimensionality, we represent the capacity of batteries by using an approximation of the average capacity. To verify that our progression of average capacity over time is close to the true average capacity of batteries in the swap station, we use two validation methods. First, we create a disaggregated MDP model with a $(M+1)$-sized state vector which captures the capacity of each individual battery. Second, we perform a Monte Carlo simulation and track the individual and average battery capacity values. For both methods, we compare the optimal expected total reward and sample paths of the optimal policy. From the results of the two methods, we are able to validate our assumption of using the average capacity as opposed to each individual battery capacity.

2.4.1.1 Disaggregated Markov Decision Process Model

In this section, we introduce a disaggregated-MDP model by capturing the capacity of each individual battery. We compare the results of the disaggregated-MDP model to the two-dimensional MDP model presented in Section 2.3. We denote the MDP model presented in Section 2.3, which has a two-dimensional state vector, as the **aggregated-MDP**. We then developed an MDP with an $(M+1)$-sized state vector where the first element represents the number of full batteries and the remaining M elements correspond to the individual battery capacity values. We denote this second MDP model as the **disaggregated-MDP**. Specifically, the state vector of the disaggregated-MDP is $s_t = (s_t^1, \vec{s}_t^{\,2})$, where $\vec{s}_t^{\,2}$ is a vector of size M corresponding to the capacity of each individual battery.

The action spaces are identical for both the aggregated and disaggregated MDPs. Specifically, the action space is two-dimensional, where the first and second dimensions denote the number of batteries to charge/discharge and replace, respectively. Specifically, the action space $A = (A^1 \times A^2)$, where $A_t^1 = \{\max\,(-s_t^1, -\Phi), \ldots, 0, \ldots, \min(M - s_t^1 - a_t^2, \Phi)\}$ and $A_t^2 = \{0, \ldots, M - s_t^1\}$ for time t. The two dimensions of these action spaces depend on Φ, s_t^1, M, and a_t^2. Φ and M represent the number of plug-ins and number of batteries at the swap station and thus, are the same for both MDPs. The first dimension of the state space is the same for both models; thus, s_t^1 impacts the action space identically for both models. The action a_t^2 represents how many batteries are replaced at time t, which is the same for both models. However, we acknowledge that it is possible for the aggregated and disaggregated MDPs to have different optimal policies even with identical action spaces because of the differences in the state space and probability transition.

For a given action, Equations (2.3) and (2.5) govern how, in the aggregated-MDP, the state transitions for the first and second dimensions, respectively. However, for the disaggregated-MDP, we must determine how a given action changes individual battery capacity values. Therefore, we must set assumptions to equate a determined number of batteries that are charged (or discharged), replaced, and swapped to actions and changes to the capacity of individual batteries. We reiterate that the action spaces for the disaggregated and aggregated MDPs are identical. Thus, as a result, we must translate a given action indicating the total number of batteries that are charged, discharged, and replaced at one time and translate this to individual batteries for the disaggregated MDP. Such assumptions are not necessary for the aggregated MDP because we model batteries as an aggregated average capacity. We emphasize that these assumptions do not fix the policy, but provide instructions on how an action impacts the individual battery capacity values. Even with these assumptions, we must still use optimization to determine which action should be taken when in each state and time. We summarize these assumptions as follows.

1. We replace the a_t^2 batteries with lowest capacity. This rule corresponds with our intuition that a swap station would want to keep batteries with higher capacity and replace those with lowest capacity.

2. We charge/discharge the a_t^1 batteries with highest capacity. This rule also corresponds with our intuition that the swap station wants to cycle through its inventory. If a battery has not been used recently at the station (has higher capacity), it is next in line to be charged/discharged. Further, this results in more battery use before any battery must be replaced.

3. After batteries are replaced, charged, and discharged, the remaining highest capacity batteries are swapped to meet as much demand as possible. This rule results in higher immediate revenue at the swap station from higher capacity swaps and hopefully higher customer satisfaction.

To be transparent, these assumptions are necessary for the disaggregated-MDP with a $(M+1)$-sized state vector. In the $(M+1)$-sized problem, we do not track the individual charge status (full/depleted) of each battery and instead track the number of full batteries. If we did track the charge and capacity of each battery, we would need to create a $2M$-sized state vector. While the MDP model with a $2M$-sized state vector is more realistic, it severely runs into the curses of dimensionality. Furthermore, a more realistic model would capture individual charge/discharge

Table 2.1: Size of the state space for the aggregated and disaggregated-MDP models.

Model	State Space Size
Aggregated-MDP	$(M+1)(\frac{1-\theta}{\varepsilon}+2)$
Disaggregated-MDP	$(M+1)(\frac{1-\theta}{\varepsilon}+2)^M$

and replacement actions for each battery which would result in a $2M$-sized action vector. As we run into the curses of dimensionality for the disaggregated-MDP, we leave for future work the investigation into a more realistic MDP with $2M$-sized state and action vectors.

However, our key point is that aggregation does not dramatically change the optimal policy and optimal expected total reward. Furthermore, aggregation allows for the benefit of solving much larger instances as it reduces the curses of dimensionality. Thus, when considering the trade-off between the difference in expected total reward vs. the ability to solve larger problems, we believe the aggregated model is a promising approach.

We proceed by showing that there is no remarkable difference between the aggregated-MDP and disaggregated-MDP models with respect to the optimal expected total rewards (objective function values) and sample paths of the optimal policies.

A policy is a set of decision rules indicating which action from the action space should be selected when in each state and time. Thus, it is possible to derive an identical policy with both models. However, it is more likely to derive different optimal policies for the two models due to the difference in the rewards. Specifically, in the aggregated model reward calculation we use the average capacity instead of individual batteries capacity. However, in the disaggregated model, we calculate the reward based on individual battery capacity values. As a result, there could be differences in the reward and resulting optimal policy. To illustrate this, consider that we have six batteries (4 full and 2 empty) and the capacity of batteries equal 0.9, 0.92, 0.94, 0.96, 0.98, and 1, which results in an average battery capacity of 0.95. If we swap one battery in the aggregated model, the revenue earned for this swap is calculated based on the average capacity, 0.95. Contrarily, in the disaggregated model, the revenue earned for this swap is calculated based on the highest battery capacity, 1. Thus, this can introduce slight differences in the reward due to

Table 2.2: Example showing the size of the state space for the aggregated and disaggregated MDP models.

Model	$M = 2$ $\varepsilon = 0.001, \theta = 0.8$	$M = 5$ $\varepsilon = 0.001, \theta = 0.8$
Aggregated-MDP	606	1212
Disaggregated-MDP	16766	$1.008 * 10^{12}$

the revenue earned per swap.

Because of the size of the problem, we are not computationally capable of solving the disaggregated-MDP with $M = 2$ batteries and capacity increment $\varepsilon = 0.001$ on a high performance computer, due to memory issues. In Table 2.1, we present the size of the state space for the aggregated and disaggregated MDPs. For the disaggregated-MDP, the size of the state space and transition probabilities is *exponential* in M (the number of batteries) which limits its applicability. In Table 2.2, we show the size of the state space for the aggregated and disaggregated MDP models when $M = 2$ and $M = 5$, to show the significance of the increase in the size of the state space for a small increase in the number of batteries.

We now demonstrate that the performance of the two models is comparable and thus, validates our use of the aggregated-MDP. For our computational tests, we are able to solve the disaggregated MDP with $M = 2$ and $\varepsilon = 0.01$. It is worthwhile to mention that we utilized 417GB of memory and 7 hours on three shared memory nodes each with quad Xeon octo-core 2.4 GHz E5-4640 processors to solve cases with $M = 2$ and $\varepsilon = 0.01$. However, increasing M by one unit to $M = 3$ requires at least 1.6TB of memory, which is beyond our available computational resources. We proceed by presenting the results for $M = 2$ and $\varepsilon = 0.01$ which show promise, but are limited due to the computational demands of the disaggregated-MDP.

Using the 40 scenarios of the Latin hypercube designed experiment described in Section 2.5.2, we computationally compare the results of the aggregated vs. disaggregated MDP models. As the capacity precision of the disaggregated MDP is $\varepsilon = 0.01$, we round all δ^C values for all 40 scenarios to the nearest multiple of $\varepsilon = 0.01$. We let the expected total reward of the aggregated and disaggregated MDPs equal $\upsilon_N^{\pi_1}(s_1)$ and $\upsilon_N^{\pi_2}(s_1)$, respectively. We calculate the percentage difference in the expected total reward of each scenario using Equation (2.15). The percentage difference informs the relative change between the objective function values (expected total rewards) of the two MDPs models.

$$\% \text{ change in the expected total reward} = \frac{\upsilon_N^{\pi_2}(s_1) - \upsilon_N^{\pi_1}(s_1)}{\upsilon_N^{\pi_1}(s_1)} * 100\%. \tag{2.15}$$

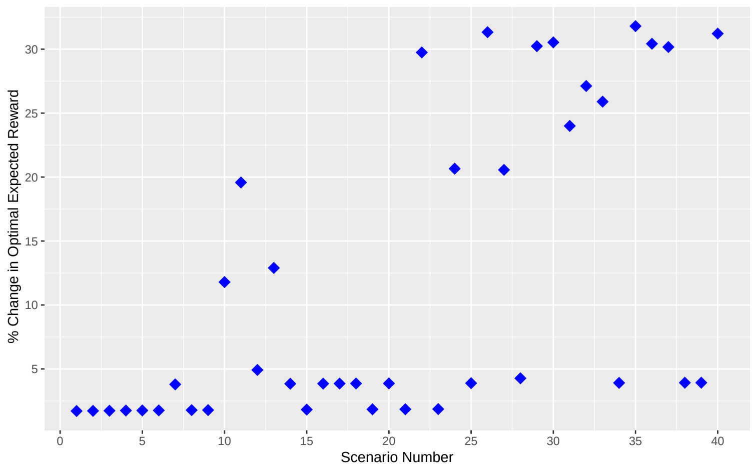


Figure 2.2: Percentage change in the optimal expected total reward of the aggregated vs. disaggregated MDP models.

In Figure 2.2, we present the percentage change in the expected total reward of each scenario. This figure demonstrates that 60% of the scenarios have lower than 5% deviation in the optimal expected total reward. The average change is 11.93%, which is consistent with generally accepted optimality gaps for approximate methods (e.g., (Zhou et al., 2020a; Novas et al., 2020; Cook and Lodree, 2017; Zhou et al., 2020b; Pérez Rivera and Mes, 2017; Luo et al., 2016)), and the maximum change is 31.8%.

Taking this one step further, we do not solely rely on the expected total reward (equivalent to objective function value) to compare the two models. As the actions of the model equate to the operations the swap station will take on a day-to-day, hour-by-hour basis, we believe comparing these actions between the disaggregated and aggregated model is necessary. Hence, we also examine the optimal policy (equivalent to solution) and illustrate how the optimal policies of the two models have small differences.

In Figures 2.3 and 2.4, we depict the sample path of the optimal policy for two scenarios when the realized demand equals the mean demand. Figure 2.3 corresponds to Scenario 5, which resulted in 1.7% change in expected total reward, and Figure 2.4 corresponds to Scenario 35, which resulted in the largest change in expected total reward, 31.8%.

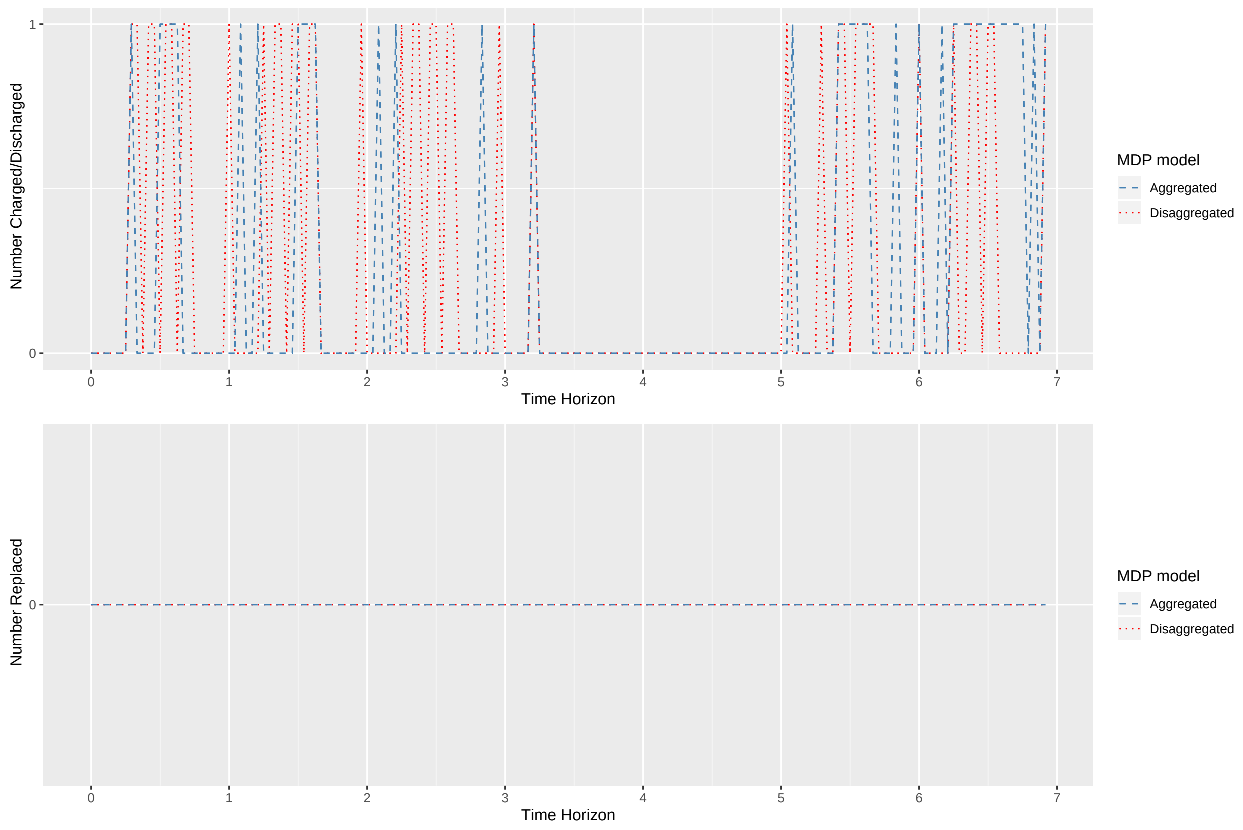


Figure 2.3: Optimal policies of the aggregated versus disaggregated MDPs for Scenario 5 when realized demand equals the mean demand.

In Figure 2.3, we observe that no replacement occurs for either MDP model. Further, both models do not indicate to discharge any batteries. We do observe that *when* batteries are charged does change slightly between the two policies. Not obvious from the figure is the total number of batteries charged over the entire time horizon for the aggregated and disaggregated models are 41 and 40, respectively. Not only are the total number of batteries charged very close, but the timing is consistent. In 124 out of 167 decision epochs, the charging actions are the same for both models.

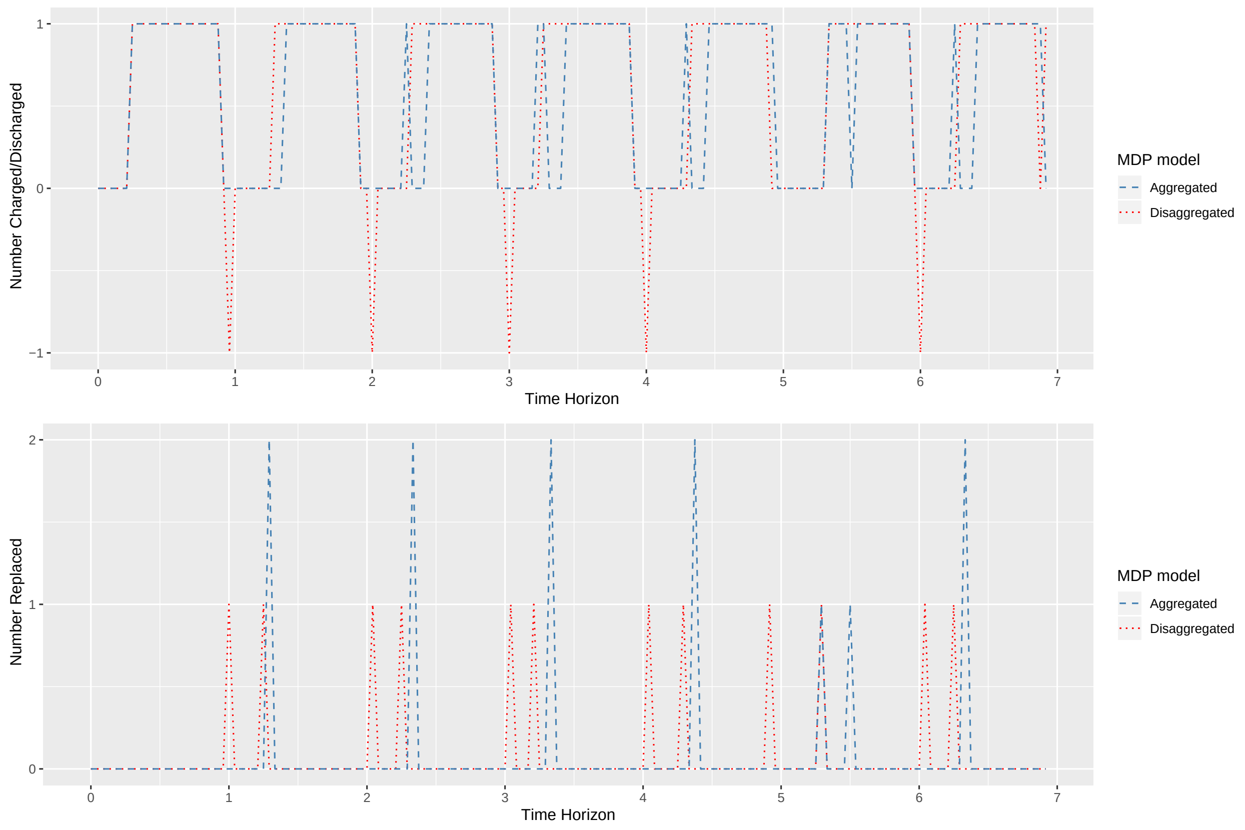


Figure 2.4: Optimal policies of the aggregated versus disaggregated MDPs for Scenario 35 when realized demand equals the mean demand.

In Figure 2.4, we depict a sample path of the optimal policy for Scenario 35. This scenario resulted in the largest change in the optimal expected total reward which could indicate that the optimal policies are different. However, when we dig into the optimal policy, we find that the optimal charging and replacement actions are similar in amount and timing. When comparing the two models, the number of decision epochs in which the optimal actions are the same are 141 charging/discharging actions and 150 replacement actions over the 167 decision epochs. Additionally, the number and timing for battery replacement is very similar between the two MDP models. In total over the entire time horizon, the policy indicates to replace 12 batteries for both the disaggregated-MDP and the aggregated-MDP model. In total, the disaggregated and aggregated MDP models charged 106 and 96 batteries, respectively. The only notable difference is that the optimal policy for the disaggregated-MDP indicates to discharge 1 battery near midnight for 5 days over the one-week time horizon, but the aggregated MDP never discharges any batteries. We emphasize that our point is not to verify that the aggregated and disaggregated models generate the same reward and policy for all cases. Instead, we seek to show that the optimal expected total reward and optimal policies under aggregation and disaggregation are close enough to justify the benefits, in reducing the curses of dimensionality, of the aggregated MDP for SAIRPs.

Hence, when we consider the trade-offs between accuracy and size of problem that we can computationally solve, we believe aggregation is a valid approach for modeling SAIRPs. We acknowledge that we cannot provide any guarantee on the percentage difference between the aggregated and disaggregated model for practical size problems because there is no computational way to optimally solve the disaggregated model. Without an optimal value for the disaggregated model, such a guarantee is not possible. We used sophisticated, high-performance computers with three shared memory nodes, quad Xeon octo-core 2.4 GHz E5-4640 processors, and 768GB of memory and yet we are only able to optimally solve the disaggregated model for two batteries. Such complexity is common among stochastic optimization problems. It is common practice when encountering such complex problems, to either make assumptions and/or use heuristic methods. We selected to make the assumption that we are not representing each battery capacity individually, and instead, we use the average battery capacity in the system. We denote this model as the aggregated model. Even with this assumption, the aggregated model is large and hard to solve and thus, we use both an exact solution method (backward induction) and two approximate solution methods (the heuristic benchmark policy and reinforcement learning approach). We could have instead just used a heuristic method for the disaggregated model. However, we reiterate that even with this approach, we would not be able to provide a guarantee on the quality of a heuristic method because there is no computational way to optimally solve the disaggregated model for practical or even small size problems.

In our analysis, we seek to show that even when faced with these computational challenges, we believe that using the aggregated model, which we can optimally solve for a larger number of batteries, is reasonable. We are not attempting to say that the disaggregated model is exactly equivalent to the aggregated model. However, instead, we seek to show that for problem sizes that we can optimally solve, the percentage difference is reasonable. To support this conclusion, we examine both the value and solutions returned. For the majority of scenarios, the percentage difference in the expected total reward (objective value) is less than 5%. Furthermore, the solution/policy is what is actually implemented by the swap station, thus, we quantify how these operations differ between the two models. The average absolute difference between the actions indicating the total number of batteries to charge, discharge, and replace over all scenarios are tiny and equal 6.4, 0.5, and 0.4, respectively. This implies that the optimal policies do not shift under the aggregation assumption.

In conclusion, we believe that the results from the disaggregated-MDP with a $(M+1)$-sized state vector show promise to indicate that aggregation of battery capacity is valid. We only see slight changes in the expected total reward and optimal policy. We believe such small changes are worth the ability to solve larger problem instances with the aggregated-MDP. However, we are transparent that these results are limited due to the computational resources necessary (memory). Thus, we next describe how we validated the aggregation of capacity for larger problem instances using a Monte Carlo simulation.

2.4.1.2 Monte Carlo Simulation

In our Monte Carlo simulation, we track both the individual status of each battery and the average capacity progression used in the presented SAIRP model. For each individual battery, we record the charge status (1 or 0) and the capacity (in $[0,1]$). We consider M individual batteries in the swap station that all start with full capacity and full charge. Then during each time period t, we uniformly at random generate the number of batteries to charge and swap demand, denote these α_t and γ_t, respectively. The α_t batteries with the highest capacity that are not full are selected for charging. The γ_t batteries with the highest capacity that are full are selected for swapping. Batteries whose capacity drops below the replacement threshold θ are selected for replacement.

Table 2.3: Differences in the simulation tracking each individual battery capacity as compared to the average capacity progression calculated in the SAIRP model.

	Absolute Error			Percentage Error		
Time Horizon	Average	Max	Min	Average	Max	Min
1 week	4.53%	15.42%	0.00%	-1.51%	8.16%	-13.09%
1 month	5.46%	15.73%	0.00%	3.43%	10.93%	-13.25%

We record the action for each individual battery and update the charge status for the start of the next time period by recording that charging batteries become full, swapped batteries become not full, replaced batteries become full, and batteries with no action stay the same. We do not consider discharging batteries as our computational results infrequently showed this action. We also update the capacity status for each individual battery using the following. Batteries selected for charging, start the next time period with δ^C less capacity. Batteries selected for replacement, start the next time period with full capacity. When batteries are swapped, the capacity of the swapped battery is unknown. Thus, batteries selected for swapping start the next time period with a capacity value selected uniformly at random from $[\theta, 1]$. Using the number of batteries that are charged, swapped, and replaced, we concurrently update the approximate average capacity that our model will use, s_t^2, in accordance with Equation (2.4).

We performed this simulation for $M = 100$, $\delta^C = 0.01$ and $\theta = 0.8$ for a time horizon of 1 week (168 hours) and 1 month (720 hours). Due to the randomly generated number of batteries charged, number of batteries swapped, and swapped capacity, we repeated the simulation 5 times for both time horizons. Upon completion of the simulation, we use the individual capacity of each battery to calculate the true average battery capacity for each time period, denote this η_t. We compare η_t to s_t^2 using an absolute difference $(\eta_t - s_t^2)$ and percentage difference $|\eta_t - s_t^2|/\eta_t$ for each time period. We present the average, minimum, and maximum absolute and percentage difference values over all simulations and time periods in Table 2.3. As is evident by the results in Table 2.3, we believe tracking the average capacity of batteries in the swap station is a good approximation in face of the computational challenges of tracking individual battery capacity.

2.4.2 Exact Solution Method: Backward Induction

We seek to solve our finite horizon Markov Decision Process (MDP) model to maximize the expected total reward given in Equation (2.11). As explained in Section 2.3, the state and action spaces of our MDP are finite, hence there exists at least one deterministic policy that is optimal (Puterman, 2005). Thus, we use backward induction to find an optimal deterministic policy for the number of batteries to charge, discharge, and replace when in each state and time. Backward induction (Puterman, 2005) is a dynamic programming algorithm that starts from the last decision epoch and steps backward in time to determine the set of best actions for the optimality Equations (2.12). The optimal policy is comprised of the set of best actions for all states and times. We note that backward induction returns the optimal policy regardless of which distribution function is used to model the uncertainty of the random variables (Puterman, 2005).

2.4.2.1 Effect of the Problem Size

The size of the stochastic SAIRP impacts the required computational time and memory for determining an optimal policy. We measure the size in terms of the number of batteries, M, the time horizon, N, the capacity increment, ε, and the replacement threshold, θ. The size of the state space is $O(\frac{M(1-\theta)}{\varepsilon})$ and the size of the action space is $O(M^2)$. This results in the size of the probability transitions to be $O(M^4N(\frac{1-\theta}{\varepsilon})^2)$ and the size of the optimal policy to be $O(MN\frac{1-\theta}{\varepsilon})$. Thus, as we consider a realistic number of batteries, a precise capacity increment, and a realistic time horizon, we run into the curses of dimensionality. The curses of dimensionality limits the problem size we can consider for computational experiments.

2.4.3 Approximate Solution Methods

In this section, we propose two approximate solution methods to address the limitations of dynamic programming and overcome the curses of dimensionality. First, in Section 2.4.3.1, we discuss our heuristic benchmark policy. Next, we present a reinforcement learning (RL) method in Section 2.4.3.2 where we use an iterative process for approximating the value function. We note, RL also goes by the term approximate dynamic programming (ADP) in the operations research community (Powell, 2011).

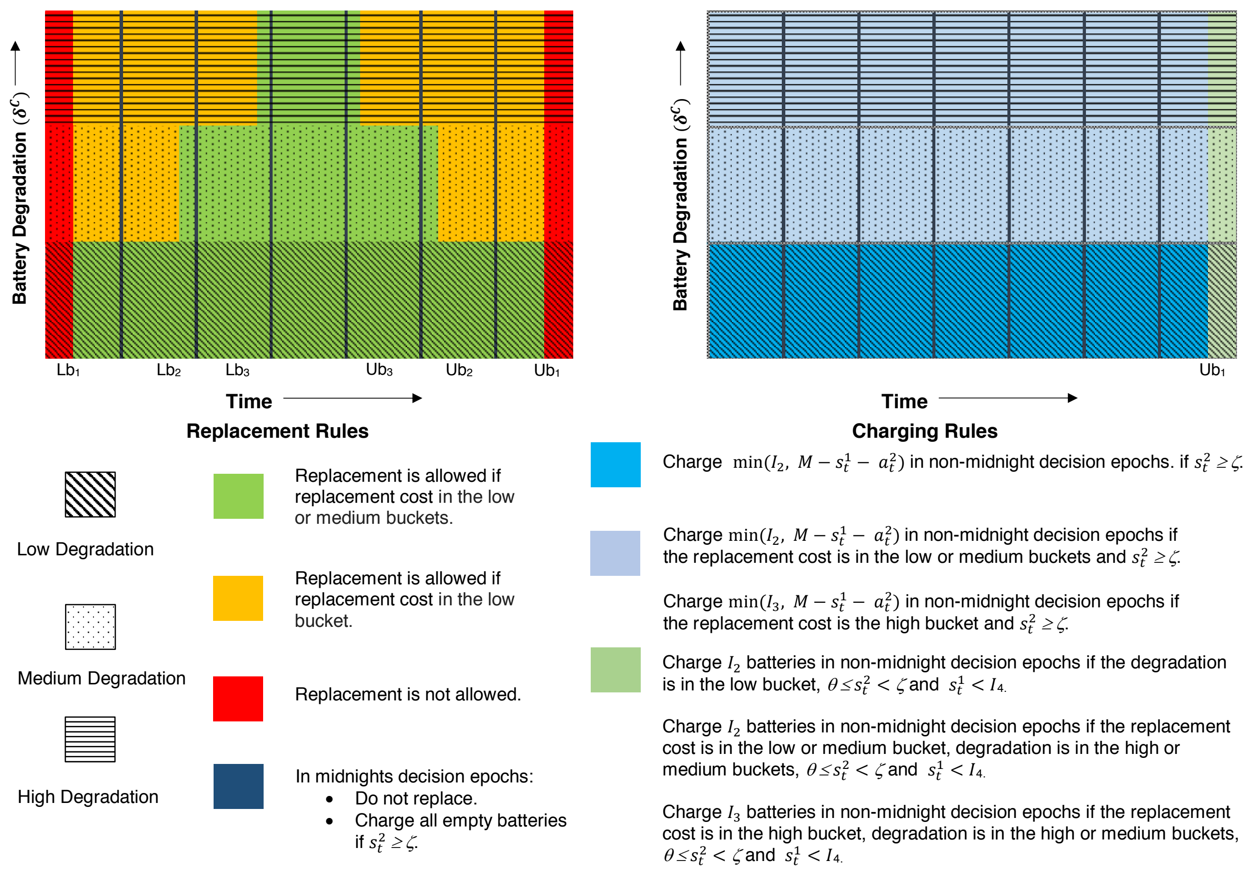
2.4.3.1 Heuristic Benchmark Policy

Although we implemented techniques to reduce the size of the MDP developed, we still run into the curses of dimensionality due to the size of the problem. Different types of policies, including myopic policies, look-ahead policies, policy function approximations, and value function approximations can be used as an approximation of the optimal policies (Powell, 2011). A myopic benchmark policy is comprised of a set of rules that dictate the action to be taken when in each state and decision epoch. Myopic policies are easy to implement and they do not rely on the forecasts based on decisions made in the future (Powell, 2011). Further, due to their simplicity, these policies are scalable for solving large problem instances. Hence, as an approximate solution method, we propose a heuristic benchmark policy, similar to a myopic policy, which is easy-to-implement, fast, and does not explicitly use forecasted information or make decisions for the future in the present.

We note that our heuristic benchmark policy is not myopic in the sense that it does not myopically optimize the problem for the present time. Specifically, the actions determined for each state and time from the heuristic benchmark policy are not necessarily the best actions from a myopic perspective, but they dynamically select actions according to the values for input parameters and the present decision epoch. Our benchmark policy constructs a solution dictating the action to take when in each state and time, in line with a constructive heuristic algorithm; this approach is consistent with other researchers (Huang and Liang, 2011; Wang and Meng, 2008; Yoon and Albert, 2020; Van der Heide et al., 2018; Yu et al., 2019; Lee et al., 2015) proposing a rule, strategy, or algorithm as their benchmark policies to solve complex optimization problems. A true myopic approach is not appropriate for our problem because short-term and long-term actions are in conflict. The actions needed to operate batteries in the short-term are exactly what is detrimental for long-term operation. Furthermore, short-term optimization might overlook the system's variability thereby having the potential to yield low-performance solutions in the long-term.

Initially, we created many different heuristic benchmark policies based on observations in the preliminary results. From these observations, we created a policy where the actions taken (i.e., the number of batteries that are charged and replaced) depend most on the replacement cost, degradation rate, and time-of-day. Our act of refining the benchmark policy is analogous to refining routines and subroutines within different heuristics (Schneider and Kirkpatrick, 2006; Zhou et al., 2020a; Wu et al., 2020; Li et al., 2020; Asghari and Al-e-hashem, 2020; Yao et al., 2020; Zhen et al., 2020; Zhong et al., 2020; Lee et al., 2015). We present the full details of this policy in Algorithm 1. To aid the communication of the policy, we also visually depict the rules in Figure 2.5. We note, although we refined this heuristic benchmark policy based on preliminary experiments, we demonstrate in Section 2.5.3 that this policy is robust and performs well for many different SAIRP instances under different conditions. Overall, in this policy, we prescribe the number of batteries that are charged and replaced based on the input state and model parameters. We proceed by explaining the logic behind the heuristic benchmark policy using replacement cost, degradation rate, and time-of-day.

First, we discuss when and under what conditions we replace batteries. Informally, we replace all depleted batteries when the average capacity is just above the replacement threshold. We disallow replacement, based on replacement cost and degradation rate, early or late in the time horizon. We define early and late in the horizon differently based on the degradation rate. As the degradation rate increases, batteries lose capacity more quickly, and more frequent large replacement costs occur. Thus, when degradation rate and replacement costs are higher, we strictly constrain the times when replacement can occur. We note that the optimal policies are computationally demanding to calculate and rely on the complex relationships between the input factors and system dynamics. Our purpose in developing the heuristic benchmark policies is to provide an easy-to-implement, well-designed, and practical approach for swap station managers. The benchmark policies do not require sophisticated optimization and instead provide a set of rules



🟩	Replacement is allowed if replacement cost in the low or medium buckets.
🟧	Replacement is allowed if replacement cost in the low bucket.
🟥	Replacement is not allowed.
🟦	In midnights decision epochs: • Do not replace. • Charge all empty batteries if $s_t^2 \geq \zeta$.

🟦	Charge $\min(I_2,\ M - s_t^1 - a_t^2)$ in non-midnight decision epochs. if $s_t^2 \geq \zeta$.
🟦	Charge $\min(I_2,\ M - s_t^1 - a_t^2)$ in non-midnight decision epochs if the replacement cost is in the low or medium buckets and $s_t^2 \geq \zeta$. Charge $\min(I_3,\ M - s_t^1 - a_t^2)$ in non-midnight decision epochs if the replacement cost is the high bucket and $s_t^2 \geq \zeta$.
🟩	Charge I_2 batteries in non-midnight decision epochs if the degradation is in the low bucket, $\theta \leq s_t^2 < \zeta$ and $s_t^1 < I_4$. Charge I_2 batteries in non-midnight decision epochs if the replacement cost is in the low or medium bucket, degradation is in the high or medium buckets, $\theta \leq s_t^2 < \zeta$ and $s_t^1 < I_4$. Charge I_3 batteries in non-midnight decision epochs if the replacement cost is in the high bucket, degradation is in the high or medium buckets, $\theta \leq s_t^2 < \zeta$ and $s_t^1 < I_4$.

Figure 2.5: Graphical representation of heuristic benchmark policy actions.

that results in an implementable policy. Furthermore, for practical, large-scale problems when other approximate methods take significant computational time, the heuristic policy provides applicable policies in seconds. Moreover, as we show in Section 2.4.3.2, the heuristic policy can be coupled with reinforcement learning methods to provide near-optimal solutions.

Formally, the replacement action depends on whether the replacement cost is high, low, or in the middle. Thus, we define κ_1 and κ_2 and construct ranges $(0, \kappa_1]$, (κ_1, κ_2), $[\kappa_2, \infty)$ for the low, medium, and high buckets of replacement cost, respectively. If the input replacement cost L_t is in the high bucket, no batteries are ever replaced. Further, if the input decision epoch is very early or very late in the time horizon, we do not allow batteries to be replaced (see red areas in Figure 2.5). If the replacement cost is in the medium or low buckets, we dictate a policy to replace all batteries during certain time periods if the average capacity (s_t^2) drops below ζ. We set

Algorithm 1 Heuristic Benchmark Policy

```
1: Input Time and State: decision epoch $t$, $s_t^1$ full batteries, $s_t^2$ average capacity
2: Input Model Parameters: degradation rate $\delta^C$, replacement cost $L_t$, and number of batteries $M$
3: Initialize: $a_t^1 \leftarrow 0$ and $a_t^2 \leftarrow 0$
4: if $s_t^2 \geq \zeta$ then                                ▷ Examine higher average capacities.
5:     if $t \% 24 = 0$ then                                  ▷ Midnight?
6:         $a_t^1 \leftarrow M - s_t^1$
7:     else if $\delta^C \leq \Delta_1$ then                  ▷ Low degradation?
8:         $a_t^1 \leftarrow min(M - s_t^1 - a_t^2, I_2)$
9:     else if $\delta^C > \Delta_1$ and $L_t \leq \kappa_2$ then   ▷ Degradation not low and replacement cost not high?
10:        $a_t^1 \leftarrow min(M - s_t^1 - a_t^2, I_2)$
11:    else if $\delta^C > \Delta_1$ and $L_t > \kappa_2$ then   ▷ Degradation not low and replacement cost high?
12:        $a_t^1 \leftarrow min(M - s_t^1 - a_t^2, I_3)$
13:    end if
14: else if $\theta \leq s_t^2 < \zeta$ and $t \% 24 \neq 0$ then   ▷ Examine lower average capacities at non midnight times.
15:    if $s_t^1 < I_4$ and $t \geq Ub_1$ then   ▷ Less than $I_4$ full batteries close to the end of time horizon?
16:        if $\delta^C \leq \Delta_1$ or ($\delta^C > \Delta_1$ and $L_t \leq \kappa_2$) then   ▷ Degradation low? Or degradation not low and
                                                       replacement cost not high?
17:            $a_t^1 \leftarrow I_2$
18:        else if $\delta^C > \Delta_1$ and $L_t > \kappa_2$ then   ▷Degradation not low and replacement cost high?
19:            $a_t^1 \leftarrow I_3$
20:        end if
21:    else if $s_t^1 < I_I$ and $Lb_1 < t < Ub_1$ then   ▷ Less than $I_I$ full batteries and not too early and late?
22:        if $\delta^C \leq \Delta_1$ then                  ▷ Low degradation?
23:            $a_t^2 \leftarrow M - s_t^1$
24:        else if $\Delta_1 < \delta^C \leq \Delta_2$ then   ▷ Medium degradation?
25:            if $L_t \leq \kappa_1$ or ($L_t > \kappa_1$ and $Lb_2 < t < Ub_2$) then   ▷ Replacement cost low? Or replacement
                                                       cost not low and not early and late?
26:                $a_t^2 \leftarrow M - s_t^1$
27:            end if
28:        else if $\delta^C > \Delta_2$ then              ▷ High degradation?
29:            if $L_t \leq \kappa_1$ or ($L_t > \kappa_1$ and $Lb_3 < t < Ub_3$) then   ▷ Replacement cost low? Or replacement
                                                       cost not low and middle of the horizon?
30:                $a_t^2 \leftarrow M - s_t^1$
31:            end if
32:        end if
33:    end if
34: end if
35: return Optimal decisions $a_t^1$, $a_t^2$ at time $t$ with current state $(s_t^1, s_t^2)$.
```

ζ greater than the replacement threshold θ to ensure that we never enter the absorbing state. The time periods we allow batteries to be replaced depends on the battery degradation rate. Similar to replacement cost, we define Δ_1 and Δ_2 and construct buckets $(0, \Delta_1]$, (Δ_1, Δ_2), $[\Delta_2, \infty)$ for the low, medium, and high degradation rate buckets, respectively. If the degradation rate is low and the replacement cost is in the low or medium buckets, we replace all batteries once the average capacity drops below ζ (see low degradation in the left portion of Figure 2.5). If the degradation rate is in the medium bucket and the replacement cost is low, we replace all batteries once the average capacity drops below ζ. Instead, if the degradation rate is in the medium bucket and the replacement cost is in the medium bucket, we replace all batteries if the average capacity drops below ζ and the time is not early or late in the time horizon (i.e., $t \nleq Lb_2$ and $t \ngeq Ub_2$). Lastly, if the degradation rate is high and the replacement cost is in the low bucket, we replace all batteries once the average capacity drops below ζ (see high degradation in the left portion of Figure 2.5). Instead, if the degradation rate is high and the replacement cost is in the medium bucket, we replace all batteries if the average capacity drops below ζ and the decision epoch is in the very middle of the time horizon (i.e., $Lb_3 < t < Ub_3$).

Next, we discuss when and under what conditions we charge batteries. There is a cyclic nature in operating the swap station due to demand patterns. Demand is lower during the night and higher during the day. Thus, we charge all empty batteries during the midnight decision epoch. Outside of midnight, we set a reorder quantity (either I_2 or I_3) and reorder this amount when the number of charged batteries drops below a set threshold. The reorder quantity and threshold depend on the current average capacity and the replacement cost. When the average battery capacity is not near the replacement threshold ($s_t^2 \geq \zeta$), we prescribe the following charging actions. If the degradation rate is low, we charge I_2 batteries or all not full batteries if the number of not full batteries is less than I_2, equivalently $\min(I_2, M - s_t^1 - a_t^2)$. We also charge $\min(I_2, M - s_t^1 - a_t^2)$ batteries when the degradation rate is in the medium or high buckets and the replacement cost is in the low or medium buckets. Instead, if the degradation rate is in the medium or high buckets and the replacement cost is in the high bucket, we charge $\min(I_3, M - s_t^1 - a_t^2)$ batteries where $I_3 < I_2$. Thus, when the degradation rate is medium or high and the replacement cost is high, we charge less batteries.

If the average capacity is near the replacement threshold ($\theta \leq s_t^2 < \zeta$), the decision epoch is at the very end of the time horizon ($t \geq Ub_1$), and less than I_4 of the batteries are full, then we charge either I_2 or I_3 batteries, where I_4 is an inventory level and I_2 and I_3 are reorder amounts. In general, $I_3 < I_2 \leq M - I_4$. We charge I_2 batteries if the degradation rate is in the low bucket or if the degradation rate is in the medium or high bucket and the replacement cost is in the low or medium bucket. When the degradation rate is medium or high and the replacement cost is high, we charge I_3 batteries. Note, our reason for charging batteries near the end of the time horizon is twofold. First, we charge batteries to meet demand and reaching the absorbing state this late in the time horizon does not decrease the expected total reward. Second, the terminal reward (see Equation (2.9)) assumes the station can swap all remaining fully charged batteries if the absorbing state is not reached. For specific details, the rules of the heuristic benchmark policy are formally described in Algorithm 1.

All of the parameter values for this benchmark policy can be set based on the input parameters of the SAIRP MDP model. For example, I_1, I_2, I_3, I_4 can be calculated as percentages of the number of batteries, M; $Lb_1, Lb_2, Lb_3, Ub_1, Ub_2, Ub_3$ can be calculated as percentages of the time horizon N; κ_1, κ_2 can be calculated as percentages of the replacement cost; $\Delta_1, \Delta_2, \zeta$ can be calculated as percentages of the ranges on δ^C; and ζ can be calculated based on θ. As we will demonstrate in our computational results, this heuristic benchmark policy can be applied to many SAIRPs with different input parameters.

2.4.3.2 Reinforcement Learning Approach

In this section, we introduce a reinforcement learning (RL) method to solve stochastic SAIRPs of different sizes. The novel feature of our RL method is that we leverage the high-performance of our heuristic policy to initialize the value function approximation. As we will demonstrate in Section 2.5.3, our RL method with heuristic policy initialization provides near-optimal solutions for modest sized problems ($M \leq 7$, $N \leq 168$) and can solve large-scale SAIRPs (i.e., $M = 100$, $N = 744$).

Two main approaches for solving problems suffering from the curses of dimensionality can be classified as value iteration (VI) and policy iteration (PI). The choice of an appropriate method highly depends on the problem characteristics (Powell, 2011). PI methods that require generating feasible policies are not appropriate for our problem as the possible combination of actions over the time horizon is large and it is non trivial to iteratively find a feasible policy in an improving direction. Hence, we adopt an approximate VI approach. In the proposed approximate VI method, we ensure that the returned policies are feasible during the iterative process. In addition to using the benchmark policy to intelligently initialize the value function approximation, we also use a double pass procedure. We denote this RL approach the double pass with heuristic policy initialization (DHPI) RL method. We will demonstrate in Section 2.5.3 that our DPHI RL method outperforms both a standard approximate VI approach and our heuristic benchmark policy alone.

Table 2.4: Notation used in the reinforcement learning algorithm.

Notation	Description
τ_1	The number of iterations in the heuristic policy initialization
τ_2	The number of core RL iterations
$\tilde{V}_t^n(s_t)$	The heuristic value of being in state s_t at time t for iteration n using the heuristic policy
$V_t^n(s_t)$	The optimal value of being in state s_t at time t for iteration n
$\overline{V}_t^n(s_t)$	The approximate value of being in state s_t at time t for iteration n
$\hat{\upsilon}_t^n(s_t)$	The observed value of state s_t at time t for iteration n
α_t^n	The step-size value at iteration n at time t

We outline the steps of our DPHI RL method in Algorithm 2 which uses the notation described in Table 2.4. The proposed algorithm has two main stages. In the first stage, we evaluate the heuristic benchmark method for τ_1 simulated sample paths, as shown in lines 1-10 of Algorithm 2. The details of this are in lines 2-8 where for every iteration, given the initial state of the system, we move forward in time using the information from the present state, a sample observation of demand, and the action taken as calculated by the heuristic benchmark policy. Then in lines 8-10, moving backward in time, we accumulate the immediate rewards to find the heuristic value of being in state s_t at time t for iteration n, $\tilde{V}_t^n(s_t)$. If a state s_t is visited over several sample paths, we use the average to calculate $\tilde{V}_t^n(s_t)$. In Equation (2.16), we calculate the heuristic value of being in the visited state $s_{t'}$ at time t'. We store the heuristic values and policies associated with all the visited states over time for all iterations and use these to initialize the second stage of Algorithm 2.

$$\tilde{V}_{t'}^{n}(s_{t'}) = \left[\sum_{t=t'}^{N-1} r_t(s_t, a_t) + r_N(s_N)\right]. \tag{2.16}$$

In stage 2 of Algorithm 2, we perform the core value iteration method with double pass procedure for τ_2 iterations. First, we initialize the value function approximation using the heuristic values for all visited states, as given in line 11. In line 12, we calculate the approximate value of being in the terminal states using the terminal reward for all iterations. Next, in lines 13-23, we perform the double pass procedure. In the forward pass, shown in lines 13-18, for each iteration, given the present state, we sample an observation of the demand and take the action that maximizes the present contribution (immediate reward) plus the last approximation of the transitioned state's value as given by Equation (2.17). When the best action is selected, the system transitions to the future state using Equations (2.3) and (2.4). Hence, in the forward pass, we update the state and decision rules in the system moving forward in time. When the algorithm selects the action for the last decision epoch, we move backward in time over the sample path created by the visited states, realized uncertainty, and taken action to update the value approximation as shown in lines 19-22. In the backward pass, the observed value is calculated using Equation (2.18) wherein $\overline{V}_{t+1}^{n-1}(s_{t+1})$ is used as the approximation of $\mathbb{E}(V_{t+1} \mid s_t, a_t)$. Given that the optimal observed value can not be lower than the heuristic value $\tilde{V}_t^n(s_t)$, we update the observed value in line 20. Ultimately, we update the present approximation using the observed value in the current iteration and previous approximation. In Equation (2.19), the step-size function α_t^n and $(1-\alpha_t^n)$ determine the weights on the current observation and previous approximations, respectively.

$$a_{s_t,t}^{n} = \underset{a_t \in A_{s_t}}{\arg\max} \left\{ r_t(s_t, a_t) + \overline{V}_{t+1}^{n-1}(s_{t+1}) \right\}. \tag{2.17}$$

$$\hat{\upsilon}_t^n(s_t) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \mathbb{E}(V_{t+1} \mid s_t, a_t) \right\}. \tag{2.18}$$

$$\overline{V}_t^n(s_t) = (1-\alpha_t^n)\overline{V}_t^{n-1}(s_t) + \alpha_t^n \hat{\upsilon}_t^n(s_t). \tag{2.19}$$

Algorithm 2 Double Pass with Heuristic Policy Initialization RL Method

```
 1: for $n = 1, \ldots, \tau_1$ do                          ▷ Start generating heuristic policy.
 2:     Select the initial state $s_1^n$.
 3:     for $t = 1, \ldots, N-1$ do
 4:         Sample an observation of the uncertainty, $D_t$.
 5:         Determine the action $a_t^n$ using heuristic policy and move to next state, $s_{t+1}^n$.
 6:     end for
 7: end for                                                 ▷ End generating heuristic policy.
 8: for $t = N, \ldots, 1$ do
 9:     Find $\tilde{V}_t^n(s_t)$    ▷ Use Equation (2.16) to find the values of visited states in the heuristic policy method.
10: end for
11: Initialize $\overline{V}_t^0(s_t)$ for $t = 1, \ldots, N-1$.    ▷ Start double pass by inputting $\tilde{V}_t^n(s_t)$ for visited states.
12: Set $\overline{V}_N^n(s) = r_N(s)$ for $s \in S$ and $n = 1, \ldots, \tau_2$.
13: for $n = 1, \ldots, \tau_2$ do                          ▷ Start the forward pass.
14:     Select initial state $s_1^n$.
15:     for $t = 1, \ldots, N-1$ do
16:         Sample an observation of the uncertainty, $D_t$.
17:         Determine the action $a_t^n$ and move to next state, $s_{t+1}^n$.
18:     end for                                             ▷ End the forward pass.
19:     for $t = N, \ldots, 1$ do                           ▷ Start the backward pass.
20:
            $$\hat{\upsilon}_t^n(s_t) = \max(\hat{\upsilon}_t^n(s_t), \tilde{V}_t^n(s_t)). \quad (2.20)$$
                                                            ▷ Update the observed value.
21:         Smooth the new observation with the previous value using Equation (2.19).
22:     end for                                             ▷ End the backward pass.
23: end for                                                 ▷ End double pass.
```

2.5 Computational Results

In this section, we outline the computational tests performed which were used to validate the proposed model and deduce insights about operating a battery swap station. We solved all test instances of the stochastic SAIRP MDP on a High Performance Computer (HPC) that has three shared memory nodes, quad Xeon octo-core 2.4 GHz E5-4640 processors, and 768GB of memory. We note that backward induction generates optimal policies while the heuristic benchmark policy and the RL method provide near-optimal solution for SAIRPs. We proceed by summarizing the data used, the Latin hypercube designed experiment performed, and the results.

2.5.1 Data

We examined operating a drone swap station for one week where each decision epoch corresponds to one hour, thus $N = 168$. A similar set of experiments could be conducted for EVs, but due to space, we focus on drone swap stations. The first decision epoch is 0:00 on Monday and the last one is 23:00 on Sunday. The one hour increment is consistent with the time needed to recharge batteries using a level 2 or 3 charger (Morrow et al., 2008; Ribbernick et al., 2015; Tesla, 2017). To calculate the time-dependent charge cost and discharge revenue, we use the 2016 historical power prices from the Capital Region, New York (National Grid, 2016). Power prices vary by time-of-year; thus, to capture a worst-case scenario when power prices are high, we use the historical prices from December 12-18 as shown in Figure 2.6. By combining the time-varying power prices with the assumption that a drone battery has 400 Wh capacity, we derive the time-varying battery charge cost and discharge revenue. These battery capacities are consistent with the DJI Spreading Wing S1000 battery (DJI, 2019). We assume the discharging revenue equals the charging costs.

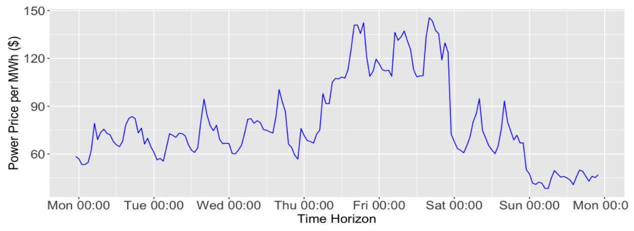


Figure 2.6: Power price fluctuation over one week.

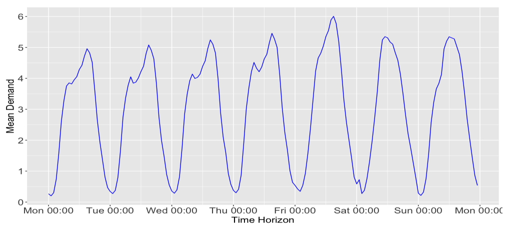


Figure 2.7: Mean demand over time.

We model the arrival of customers to a swap station using a non-homogenous (i.e., dynamically changing, with peaks and off-peaks) Poisson process. This modeling decision is consistent with many other papers modeling the arrival of customers (Widrick et al., 2018; Nurre et al., 2014; Tan et al., 2014; Sun et al., 2019; Swartzman, 1970; Green et al., 2007; Harris et al., 1987). In absence of real data representing the arrival of customers to a swap station, we assume the mean arrival of customers mimics historical observations at a gas station. This assumption is consistent with other research examining swap station arrivals (see, (Widrick et al., 2018; Nurre et al., 2014; Sun et al., 2019)). Assuming swap demand follows a Poisson distribution, we set the time-varying mean swap demand, λ_t, equal to the historical time-varying demand for Chevron gas stations Nexant, Inc. et al. (2008). In Figure 2.7, we display the mean swap demand for each decision epoch (hour) of the one-week time horizon, which has peaks and off-peaks over time. Note, the demand values can be scaled up or down based on different assumed number of customers visiting the swap station in a one week time period. To ensure that swap stations adhere to a minimum service level, we set the battery replacement threshold to $\theta = 80\%$ thereby forcing batteries to be replaced before the average capacity drops below 80%. This 80% threshold is consistent with battery end-of-life (Wood et al., 2011; Debnath et al., 2014).

We proceed by performing a Latin hypercube designed experiment and analyze how changes in input parameters impact the expected total reward, met demand, and optimal policies.

2.5.2 Latin Hypercube Designed Experiment

In this section, we summarize the design and results from performing a 40-scenario Latin hypercube designed experiment. A Latin hypercube designed experiment is a space-filling design commonly used for computer simulations (Montgomery, 2008). For each scenario of our designed experiment, we use backward induction to find the optimal policy when maximizing the expected total reward. In addition to capturing the expected total reward, we examine the charging/discharging and replacement actions in the optimal policies and calculate the amount and expected percentage of met demand. For all scenarios, we consider a drone battery swap station with $M = 7$ full-capacity batteries and discretize the average capacity in increments of $\varepsilon = 0.001$. Note, these are the largest and smallest values for M and ε, respectively, which are solvable on the high performance computer due to the curses of dimensionality. On average, the scenarios require 830GB of memory and take 3.8 hours to solve.

In the designed experiment, we change 3 factors corresponding to the revenue per swap, β, the replacement cost, L_t, and the battery degradation factor, δ^C. We select these factors as they influence how frequently batteries are replaced and the amount of demand the swap station can meet. Further, as battery technology advances, these factors will change and thus, our intent in the designed experiment is to analyze the swap station operations under different (and possibly future) conditions. For each factor, we define low and high values as outlined in Table 2.5. We set the low and high values per battery swap to \$1 and \$3, respectively. Based on current data for the price per kWh, we set the cost to replace a drone battery equal to \$2 to \$100, which represents an optimistic future forecast and todays cost (Romm, 2017). We assume the battery degradation per cycle, δ^C, is linear, which is consistent with research on the degradation for the first 500 cycles (Lacey et al., 2013; Ribbernick et al., 2015; Xu et al., 2018). We acknowledge that there are many factors which influence battery degradation and many studies that observe different degradation rates (Plett, 2011; Abe et al., 2012; Dubarry et al., 2011; Ribbernick et al., 2015). Synthesizing these studies, we use optimistic and pessimistic degradation rates by setting the low and high values of δ^C to 0.5% and 2% per cycle, respectively.

From these low and high values, we use the LHS DOE generator version 1.0.0.0 in MATLAB (Frederic, 2016) to generate 40 scenarios which cover the experiment space. We present the factors for all 40 scenarios and the results representing the expected total reward, expected met demand percentage, expected number of batteries replaced, expected number of batteries charged, and expected number of batteries discharged in Table 2.6.

Using backward induction, we derive the optimal action for every state over time. With the optimal action, we can generate sample paths of actions, states, and demand over time by assuming an initial starting state, generating a realized demand, and implementing the optimal action. The future states of the sample path are then generated using Equations (2.3) and (2.5). As there are many sequences of realized demand and corresponding sample paths, we select to display the sample path when the initial state has all batteries with full capacity and full charge and the realized demand equals the mean demand.

Using the results in Table 2.6 and the generated sample paths, we deduce insights about the operation of a drone battery swap station. First, focusing on the expected met demand, scenarios 10, 22, 24, 26, 29, 31, 35, 36, 37, and 40 all satisfy more than 80% of the demand. These scenar-

Table 2.5: Factors with associated low and high values for use in the Latin hypercube designed experiment.

Factor	Low	High
Basic revenue per swap (β)	1	3
Replacement cost L_t	2	100
Battery degradation factor (δ^C)	0.005	0.02

ios all have a higher revenue per battery swap. We reiterate that we set all swap revenue values such that they exceed the maximum charging cost. However, the results indicate that solely setting the swap revenue higher than the charging cost is not sufficient to meet all demand. Instead, the swap revenue needs to be higher to account for the future replacement costs due to battery degradation. Thus, if the replacement cost is too high in comparison to the swap revenue, the swap station is not motivated to meet demand as meeting demand causes more frequent battery replacement. However, when the prices are in line, the swap station will replace batteries and satisfy demand with batteries that have a higher average capacity.

Next, we seek to deduce insights about characteristics of the optimal policy. First, we display the results from scenario 31 where 80% of demand is met and comment on how the optimal policies for other scenarios differ. Assuming the realized demand equals the mean demand, we display sample paths of the optimal action over time (Figure 2.8), the state over time (Figure 2.9), and the met demand over time (Figure 2.10). We note, this is one of many different sample paths and scenarios examined. In both the optimal state and optimal action sample paths, we see a relatively consistent structure in terms of the number of full batteries, average battery capacity, number of batteries charging, and number of batteries replaced. Over the 7 days, approximately $1/2$ of the batteries are kept full. To keep the batteries full, we see a consistent charging pattern by day of the week. Due to the charging, we see a steady decline in the average battery capacity (see Figure 2.9). Once the battery capacity drops to 80%, the optimal replacement action for this scenario is to replace all 7 batteries, which occurs approximately every 1.5 to 2 days. This restores all batteries to full and 100% capacity. The replacement is necessary to continue to meet

Table 2.6: Scenarios and results of the backward induction algorithm for the Latin hypercube designed experiment.

Scenario	β	L_t	δ^C	Opt. Exp. Tot. Rew. ($)	Exp. Met Dem. (%)	Num. Replaced	Num. Charged	Num. Disch.	Num. Met Dem.
1	1.03	45	0.009	306.27	39.2	0	192	0	196
2	1.08	82	0.011	226.44	27.4	0	133	0	137
3	1.13	61	0.005	610.78	66.6	0	332	0	333
4	1.20	69	0.009	357.77	39.2	0	192	0	196
5	1.23	89	0.008	379.30	41.0	0	200	0	205
6	1.26	47	0.010	369.39	38.2	0	187	0	191
7	1.32	98	0.019	173.69	17.0	0	80	0	85
8	1.39	94	0.011	292.13	27.8	0	133	0	139
9	1.41	87	0.014	227.79	21.2	0	100	0	106
10	1.48	36	0.006	643.87	81.8	6	402	0	409
11	1.53	68	0.016	247.36	21.2	0	100	0	106
12	1.56	28	0.018	218.60	51.4	14	238	0	257
13	1.6	57	0.010	470.59	38.2	0	187	0	191
14	1.68	31	0.017	305.66	40.4	7	191	0	202
15	1.71	62	0.006	691.36	54.2	0	266	0	271
16	1.76	44	0.017	284.95	21.2	0	100	0	106
17	1.83	55	0.016	296.39	21.2	0	100	0	106
18	1.88	51	0.019	248.14	17.0	0	80	0	85
19	1.94	66	0.012	409.05	27.8	0	133	0	139
20	1.95	73	0.019	257.45	17.0	0	80	0	85
21	2.00	83	0.012	421.81	27.8	0	133	0	139
22	2.07	20	0.013	817.62	81.0	19	383	0	405
23	2.15	90	0.013	391.04	24.0	0	114	0	120
24	2.16	41	0.007	967.02	81.4	7	397	0	407
25	2.22	79	0.015	360.13	21.2	0	100	0	106
26	2.27	8	0.007	1351.20	82.4	21	399	9	412
27	2.32	25	0.017	749.91	76.2	22	356	0	381
28	2.37	74	0.014	384.65	21.2	0	100	0	106
29	2.40	7	0.018	1240.81	82.4	42	381	12	412
30	2.50	14	0.008	1376.01	82.2	18	398	6	411
31	2.51	24	0.015	942.54	80.0	21	377	0	400
32	2.56	34	0.009	1114.34	76.0	9	369	0	380
33	2.64	39	0.010	1051.63	79.8	12	386	0	399
34	2.68	54	0.016	469.07	41.0	0	195	0	205
35	2.71	3	0.007	1796.43	82.4	41	391	21	412
36	2.77	16	0.012	1358.72	81.8	21	391	4	409
37	2.83	21	0.013	1244.51	80.8	21	382	2	404
38	2.90	77	0.015	471.28	21.2	0	100	0	106
39	2.91	96	0.020	349.03	15.6	0	72	0	78
40	2.96	12	0.005	1836.53	82.4	7	403	14	412

demand. Note, due to the curses of dimensionality and the computational limitations of being able to only examine a one week time horizon, we consider pessimistic battery degradation rates, which scenario 31 has. Pessimistic degradation rates can result in more frequent battery replacements. In reality, batteries with a slower degradation would not need to be replaced as frequently as our results indicate. However, in spite of the computational limitations, we believe the struc-

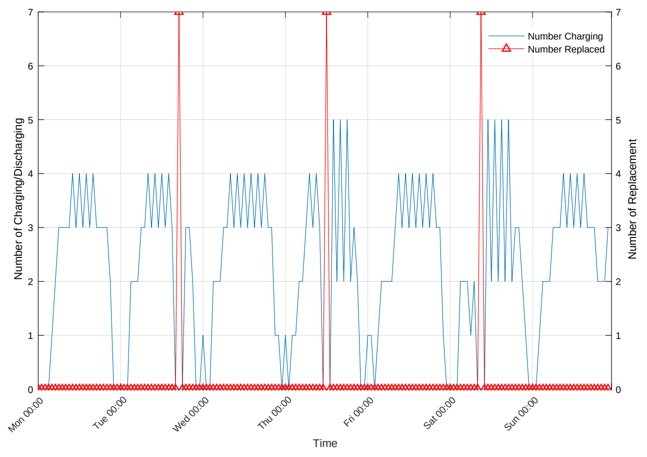


Figure 2.8: Action taken over time for a sample path of the optimal policy of Scenario 31 when realized demand equals the mean demand.

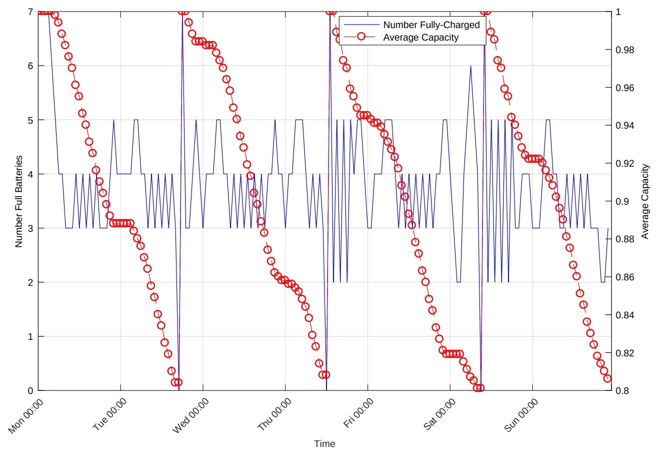


Figure 2.9: Number of full batteries and average capacity for the optimal policy over time for Scenario 31 when realized demand equals the mean demand.

ture and pattern of the optimal policy will be close to swap stations run for longer time periods and a slower battery degradation rate. In Figure 2.10, we display the demand and met demand over time. Overall, we are able to meet 80% of demand, but fail to meet all demand during high peak time periods.

When reviewing the optimal replacement policy for scenario 31, we see that when the average capacity drops to 80% the swap station replaces all batteries. This policy is present in most scenarios, however, not all. For example, in Figure 2.11 we display the sample path of the optimal state over time for scenario 22. Consistent with the results previously presented, this is a sample path when the realized demand equals the mean demand. Focusing on the changes to the

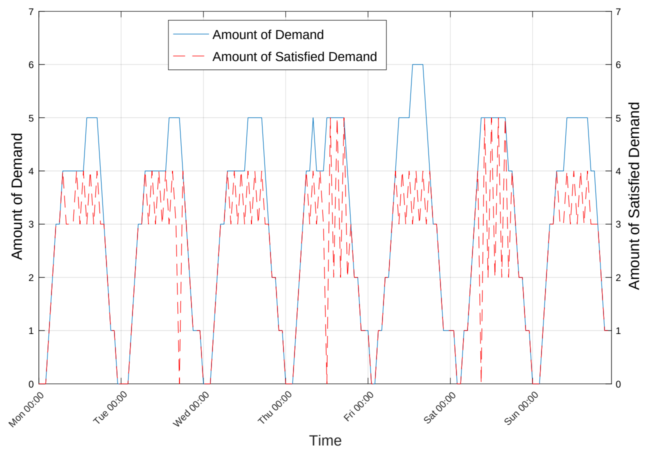


Figure 2.10: Demand and met demand for the optimal policy for scenario 31 when realized demand equals the mean demand.

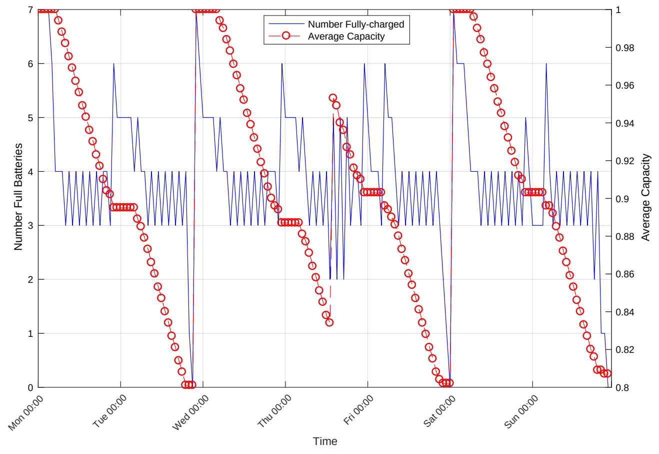


Figure 2.11: Number of full batteries and average capacity for the optimal policy for scenario 22 when realized demand equals the mean demand.

average capacity, we see that sometimes all batteries are replaced (i.e., average capacity goes up to 100%), but during mid-day on Thursday only a portion of the batteries were replaced.

The sample path for the optimal action in scenario 31 (see Figure 2.8) does not indicate to discharge any batteries. However, in seven scenarios we did observe discharging actions which earns the swap station revenue from the energy supplied. We note, in all seven scenarios with discharging, replacement also occurs. As discharging causes batteries to degrade, this behavior coincides with our intuition that when discharging occurs replacement will also occur. Further, when discharging occurs, full batteries are not available for meeting swap demand until they are recharged. However, discharging can occur during time periods with low demand without sacri-

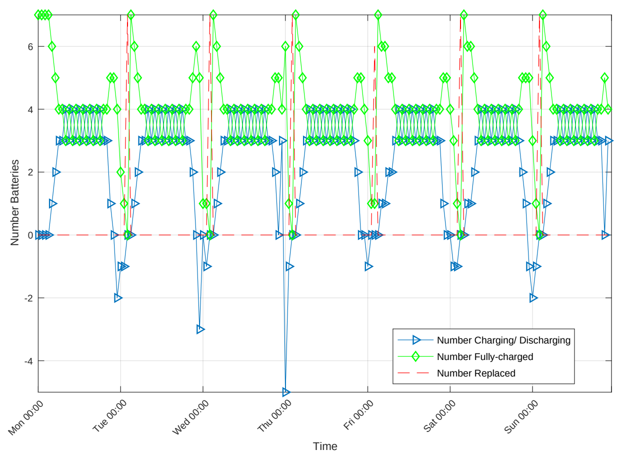


Figure 2.12: Number of full batteries and charging/discharging actions for the optimal policy over time for Scenario 35 when realized demand equals the mean demand.

ficing the satisfaction of demand. In Figure 2.12, we display a sample path for the optimal action of scenario 35 which has discharging. The discharging occurs when the swap station has a large number of full batteries during the early hours of each day when the mean demand is low.

To obtain additional insights on the effect of the three factors, we perform a one-by-one correlation analysis as summarized in Figure 2.13. Each column of Figure 2.13 corresponds to the three factors: revenue per swap, replacement cost, and battery degradation. Each row of Figure 2.13 corresponds to four different metrics: the number of batteries replaced, the expected total reward, the amount of satisfied demand, and the number of batteries charging. Each point corresponds to a scenario and is calculated based on the sample path of the optimal policy when realized demands equals mean demand.

First, we examine the number replaced as a function of the three factors. The replacement cost has the greatest impact on the number replaced in which we see zero replacements when the replacement cost is above \$41. For costs below \$41, we see a greater number of replacements as the cost decreases. We would assume that high revenue per swap values and high battery degradation values would result in a high number of battery replacements. We do see a positive correlation between the number of replacements and the revenue per swap. We justify this relationship because a higher revenue per swap incentivizes the swap station to charge batteries to meet as much demand as possible. This charging leads to battery degradation and future necessary re-

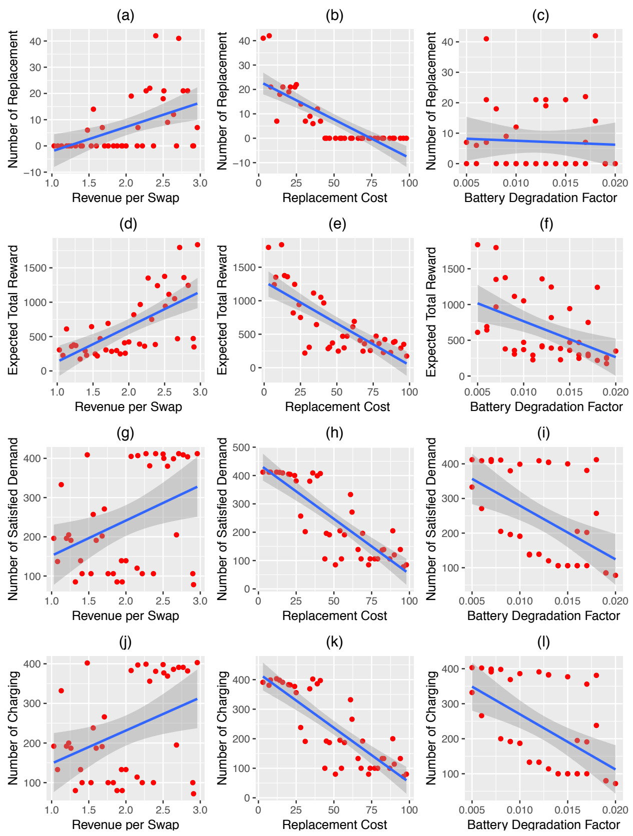


Figure 2.13: Correlation between the three factors and the number of batteries replaced, the optimal expected total reward, the amount of demand satisfied, and the number of batteries charging under the optimal policy.

placements. Thus, when the revenue per swap is high enough it offsets the future replacement costs. We observe no correlation between the battery degradation factor and the number of batteries replaced. This result is counterintuitive and indicates that replacement is not solely based on how quickly batteries degrade and instead involves a complex relationship with many factors. Next, we examine the expected total reward. The relationships are intuitive; the results indicate the expected total reward to increase as revenue per swap increases, replacement cost decreases, or the battery degradation factor decreases.

Lastly, due to their similarities, we examine both the amount of satisfied demand and the number of batteries charging. As the revenue per swap increases, we meet more demand and charge more batteries. This coincides with our intuition that when the revenue is set high enough, the swap station performs the necessary charging actions to meet demand. However, there are three scenarios (34, 38, and 39) that despite having high revenue per swap values, a relatively low amount of demand is satisfied. All three scenarios have a high replacement cost and the optimal policy indicates to not replace any batteries. We also observe two scenarios (3 and 10) with low revenue per swap and a large amount of met demand. In both cases, the battery degradation factor is small which causes the average capacity to reduce at a slow rate. In spite of this small battery degradation factor, battery replacement occurs in scenario 10 due to the small battery replacement cost. These replacements enable consistent charging and satisfaction of more than 80% of the demand. In parts (h) and (k) of Figure 2.13, we observe a negative correlation between the replacement cost and both the amount of satisfied demand and the number of batteries charged. In parts (i) and (l) of Figure 2.13, there is more occurring with each scenario than what is portrayed in the graphs. In all scenarios above the blue trend line, batteries are replaced which enables the satisfaction of demand through battery charging even when the battery degradation factor increases. Contrarily, in all scenarios below the blue trend line, batteries are not replaced and thus, less demand is satisfied thereby requiring less battery charging. When no batteries are replaced, the battery degradation factor has a greater impact on both the amount of met demand and the number of batteries charging.

2.5.3 Performance of the Approximate Solution Methods

In this section, we evaluate the performance of our two approximate solution methods through a comparison with the exact algorithm, backward induction (BI). In Section 2.5.3.1, we outline our methods for selecting the best parameters for our proposed approximate solution methods. In Section 2.5.3.2, we demonstrate the high performance of the heuristic benchmark policy and our Double Pass with Heuristic Policy Initialization (DPHI) RL method. We discuss the robustness of our heuristic benchmark policy over changes in the set of Latin hypercube designed experiment and the number of batteries. Lastly, we demonstrate the ability of the DPHI RL method to solve the large-scale SAIRPs.

Table 2.7: Parameters used for the heuristic benchmark policy.

Parameter	Value	Parameter	Value	Parameter	Value
I_1	0.25M	κ_1	$0.25L_t^{max}$	κ_2	$0.41L_t^{max}$
I_2	0.5M	Lb_1	0.05N	Ub_1	0.95N
I_3	0.3M	Lb_2	0.25N	Ub_2	0.75N
I_4	0.5M	Lb_3	0.4N	Ub_3	0.6N
Δ_1	$\delta_{min}^C + \frac{1}{3}(\delta_{max}^C - \delta_{min}^C)$	Δ_2	$\delta_{max}^C - \frac{1}{3}(\delta_{max}^C - \delta_{min}^C)$	ζ	$\theta + 0.02$

2.5.3.1 Parameter Specifics for the Approximate Methods

In this section, we describe the specific parameter values that we use for our two approximate solution methods. In our heuristic benchmark policy, there are 15 parameters. Thus, optimizing the parameter values is not a trivial task. Hence, we empirically examine different parameter values, which is consistent with the selection of heuristic and metaheuristic parameters and subroutines (Zhou et al., 2020a; Wu et al., 2020; Li et al., 2020; Asghari and Al-e-hashem, 2020; Yao et al., 2020; Zhen et al., 2020; Zhong et al., 2020; Lee et al., 2015). Specifically, to select the parameters, we test 18 sets of candidate values for the 15 parameters on the 40 instances of LHS designed experiments. In Table 2.7, we display the set of parameters that we found to produce the best overall results. To validate the selection of these parameters, we test their performance on new, distinct sets of LHS designed experiment scenarios where the number of batteries, the revenue per swap, the replacement cost, and the battery degradation factor are changed. We will demonstrate in this section the robust performance of the heuristic benchmark policy for all of these instances.

We note that a complicated combination of factors impacts the optimal policy over time, so we do not take the approach of optimizing the problem over a myopic horizon to derive the values of parameters. Instead, we design the parameters to be easy-to-implement for managers to yield robust decision rules over various scenarios. To determine the values of these parameters, we ran many preliminary experiments and observed trends and insights from sample paths of the optimal policies. We used these trends and insights to empirically test many different values and combinations of parameter values. One trend that we noticed is that replacement cost is a critical factor when deciding whether to replace batteries or not (see Figure 2.13). Thus, we define L_t^{max} to equal the present replacement cost of a battery which also is used as the upper bound for the designed experiment. We then examine battery replacement costs lower than this value because there is a forecasted decreasing trend in battery prices over time (Romm, 2017).

We use L_t^{max} to set κ_1 and κ_2 for the heuristic benchmark policy to create low, medium, and high buckets of replacement costs. These buckets allow us to analyze and determine swap stations operations if there is a dramatic or moderate decrease in battery replacement costs. When examining our preliminary experiments, we found that 100% of the scenarios with replacement cost higher than $0.4L_t^{max}$ incorporate no replacement action. When the replacement cost is less than or equal $0.4L_t^{max}$ and the revenue per swap is high enough, we observe replacement actions. Thus, we set $\kappa_2 = 0.41L_t^{max}$ to define the high bucket. From further empirical experimentation, we observed a meaningful difference in the total number of batteries replaced when the replacement cost is above or below $0.25L_t^{max}$. Specifically, on average, we replaced 9.17 when the replacement cost is between $0.25L_t^{max}$ and $0.41L_t^{max}$ and replaced 23.30 batteries when the replacement cost is below $0.25L_t^{max}$. Thus, we set $\kappa_1 = 0.25L_t^{max}$.

Next, we describe how we set the replacement threshold for the heuristic benchmark policy, ζ. As we do not want the system to enter the absorbing state and discontinue operation (when the average capacity is below θ), we set $\zeta > \theta$. Further, from preliminary experiments, we observed that of the scenarios that replace batteries, 75% have replacement occur when the average battery capacity drops below $\theta + 0.02$. As follows, we set $\zeta = \theta + 0.02$ and only allow battery replacement when the average capacity is between θ and ζ. Additionally, we restrict when replacement may occur based on time and the battery degradation factor, δ^C. Specifically, we disallow battery replacement very early and late in the time horizon of the problem. During the middle of the time horizon, we allow set intervals (LB_1, UB_1), (LB_2, UB_2), and (LB_3, UB_3) which when combined with replacement cost and the degradation factor levels (defined by Δ_1 and Δ_2) dictates when replacement can or cannot occur. We set these intervals based on preliminary experiments and noticed that only 5% of the scenarios include replacement actions outside of these intervals.

Lastly, we examined when and how many batteries are charged from our preliminary experiments. From these experiments, we observed that solely looking at how many batteries are charged gave an incomplete picture. Instead, we found that we must also examine a charge-up-to level indicating how many batteries should be full over time. Thus, we set values for (I_2 and I_3) to indicate how many batteries should be charged and (I_1, I_4) to indicate how many batteries should be full in inventory. As these values strongly interact together, we empirically examined many combinations and selected the combination that resulted in the highest overall performance. We also observe a small correlation between the number of batteries to charge and the battery degradation factor. However, we observed this to be the weakest factor when deciding the number of batteries to charge. As a result, we break down the battery degradation factor into three equal low, medium, and high buckets with Δ_1 and Δ_2. We use these buckets combined with the replacement cost buckets to calculate how many batteries to charge.

Now, we present the parameter values used for our DHPI RL method. We experimentally determined these values and selected the ones which resulted in the best convergence behavior over all scenarios. We set the number of iterations to generate sample paths of demand for the heuristic benchmark and core double pass algorithm equal to $\tau_1 = 1000$ and $\tau_2 = 100000$, respectively. We use the harmonic step-size function, $\alpha_t^n = w/(w+n-1)$, as is consistent with the literature (Powell, 2011; Rettke et al., 2016; Meissner and Senicheva, 2018), with $w = 4000$.

In Section 2.5.3.2, we compare our DPHI RL method with a standard approximated value iteration (AVI) method wherein the action selection and value approximations are performed moving forward in time. We note, as compared to our DPHI RL method, no intelligent initialization is used. We observe that adding the exploration feature, generating more sample paths of demand per iteration, and setting a higher number of iterations, τ, improves the performance of the AVI method. In our AVI method, we use the exploration feature where an action that does not necessarily maximize the summation of the present and future contributions can be selected. In the for loop of action at time t and iteration n, with our exploration feature we allow an action with a lower total contribution to replace another action with higher total contribution with a probability equal to p^n. In the AVI method, we use a linear function $p^n = 0.2 - 0.2(^n/_\tau)$ to reduce exploration as n, the iteration counter, increases. We generate 30 sample paths of demand per iteration and set the number of iterations to be $\tau = 1000000$. We use the harmonic step-size and scale w to be in line with τ, i.e., $w = 40000$.

2.5.3.2 Analysis of the Experimental Results

In this section, we provide the results of solving the designed experiment scenarios using our proposed approximate solution methods, the heuristic benchmark policy and Double Pass with Heuristic Policy Initialization (DPHI) RL. We compare the results with Backward Induction (BI) and an approximated value iteration (AVI) method. We will demonstrate the superiority of our DPHI RL method and the speed and competitiveness of our heuristic benchmark policy. Moreover, we discuss that the heuristic benchmark policy is robust to changes in the number of batteries and set of scenarios (i.e., SAIRP instances). Finally, we demonstrate the capability of our DPHI RL method for solving large-scale SAIRPs.

We solved all 40 scenarios of the designed experiment (see Section 2.5.2) using the heuristic benchmark policy, DPHI RL, AVI, and BI for $M = 7$. We note, due to the curses of dimensionality, we are not able to determine the optimal expected total reward for values higher than $M = 7$. We empirically demonstrate that the policies generated by the heuristic benchmark policy and DPHI RL are near-optimal for many different possible scenarios. We summarize the optimality gap of all scenarios using the approximate methods in Table 2.8. We calculate the optimality gap for the approximate methods using Equation (2.21) and the demand gap using Equation (2.22). To calculate the demand gap, we use the sample path where the realized demand equals the mean demand.

$$\text{Optimality Gap} = \frac{\text{BI Expected Reward - Approx. Expected Reward}}{\text{Expected Reward BI}} * 100\%. \tag{2.21}$$

$$\text{Demand Gap} = \frac{\text{BI Satisfied Demand - Approx. Satisfied Demand}}{\text{BI Satisfied Demand}} * 100\%. \qquad (2.22)$$

From the results, the average optimality gap of the heuristic benchmark policy over all scenarios is 10.03% where 45% of the scenarios have an optimality gap less than 5%. Our DPHI RL method results in the smallest average optimality gap equal to 7.35%. We note, both of these are significantly lower than 24.22%, the AVI average optimality gap.

The average computation time to generate a policy for the heuristic benchmark policy, DPHI RL, AVI, and BI are 0.01, 80.55, 660.55, and 13649.92 seconds, respectively. We note, the computational time of our heuristic benchmark policy consists of two operations: policy generation and policy evaluation. We denote the time it takes to calculate the action the swap station should take when in each state and time, using Algorithm 1, as the policy generation time. The average policy generation time over all scenarios is 0.01 seconds. We note that solely generating a policy does not give us an indication of the how well it performs. To calculate the expected reward for the heuristic benchmark policy, we use the transition probability of transferring from a current state to a future state given the action. Therefore, we do not have perfect information and instead must incorporate the uncertain demand into this calculation. Thus, we must perform a procedure analogous to backward induction to calculate the expected total reward for a given policy. As a result, the average policy evaluation time is 13360.57 for the benchmark policy.

Overall, we observe the high performance of the benchmark policy and DHPI RL Method. The benchmark policy takes fractions of a second to generate a solution. Thus, operationally, the swap station needs to take less than a second, on average, to generate a policy that they can implement to achieve this high performance in a highly variable environment having non-stationary demand for swapping with peaks and off-peaks. If the swap station is looking for a policy with a lower optimality gap, the DHPI RL method can be used and takes under 5 minutes on average.

Next, we focus on the amount of demand met using the approximate solution methods as

Table 2.8: Performance comparison of the approximate solution methods.

	Optimality Gap (%)		
Scenario	Heuristic Benchmark	DHPI RL	AVI
1	20.52	18.62	26.66
2	3.20	1.68	15.48
3	22.98	20.96	26.58
4	20.52	18.85	27.15
5	11.42	9.17	16.61
6	25.37	23.51	29.60
7	2.73	1.82	15.53
8	3.07	1.60	14.54
9	1.68	0.78	11.09
10	2.37	0.51	26.34
11	19.03	14.30	20.85
12	17.20	14.22	18.99
13	19.12	23.44	29.62
14	27.77	25.41	31.88
15	25.04	14.94	21.67
16	19.04	15.59	23.68
17	19.02	14.45	20.28
18	2.65	1.67	16.61
19	3.08	1.90	19.13
20	2.65	1.77	16.86
21	3.07	1.75	19.50
22	2.95	0.92	38.35
23	12.27	10.63	16.74
24	3.43	0.47	31.69
25	1.70	0.77	15.41
26	7.50	5.07	23.62
27	12.45	7.92	52.68
28	1.63	0.74	10.61
29	5.78	1.25	34.57
30	6.70	1.67	23.78
31	3.55	1.05	42.18
32	5.94	2.46	31.45
33	3.74	0.87	36.05
34	24.84	20.63	26.13
35	14.03	10.62	24.52
36	4.91	0.17	30.38
37	2.22	0.61	35.04
38	1.69	0.71	14.34
39	8.07	6.78	14.28
40	6.36	3.85	18.38
Avg. Gap (%)	10.03	7.60	24.22
Avg. Time (s)	0.01	80.55	660.55

compared to the optimal policy. Overall, there are 10 and 16 scenarios where the benchmark and DPHI RL solutions satisfy more demand than the optimal policy, respectively. For the heuristic benchmark policy, 67.5% of the scenarios have a demand gap less than 10%. In 40% of the scenarios, we see a demand gap of less than 10% for the DPHI RL method. On average, the amount of satisfied demand with the benchmark policy and the DPHI RL method are only 7.4% and 3% less than the amount of satisfied demand under the optimal policy, respectively.

By taking all scenarios into consideration, we can conclude that our heuristic benchmark policy and DPHI RL method are capable of generating near-optimal solutions quickly. The benchmark policy requires only bytes of memory to generate a policy which is remarkably lower than the 830GB of memory needed for backward induction. The required memory for the DPHI RL method and AVI is 1.2GB and 36GB, respectively, demonstrating RL's power to address memory issues. The memory requirements combined with the speed demonstrate the applicability of the heuristic benchmark and the DPHI RL method for solving large problem instances that backward induction is not capable of solving.

Next, we demonstrate that the selected parameters for the heuristic benchmark policy based on percentages of the SAIRP input parameters work well for a station with different number of batteries. Due to curses of dimensionality, we focus on validating the use of the benchmark policy for $M \leq 7$. Specifically, we calculate the optimal expected total reward for all 40 scenarios of the designed experiment with $M = 4, 5,$ and 6. The average optimality gap for $M = 4, 5,$ and 6 are equal to 8.61%, 8.12%, and 9.45%, respectively. These results further confirm that the benchmark policy is a suitable method for SAIRP instances with different input parameters. Thus, we can conclude that the heuristic benchmark policy is robust and capable of generating quality solutions for many different empirical tests.

The parameters for the heuristic benchmark policy outlined in Table 2.7 were determine based on many empirical observations from the tests on the 40 scenarios (see Table 2.6) with $M = 7$. Thus, to verify that these parameters perform well for other scenarios, we generated an additional 40 new scenarios and show that the benchmark policy still performs well for these new scenarios. With each new scenario, we used backward induction to calculate the optimal expected total reward and compared this to the expected total reward returned from the benchmark policy with $M = 7$. Our results indicates that the average gap between backward induction and the heuristic benchmark policy for these new scenarios is 10.3%. The small gap indicates that the high performance of the heuristic benchmark policy does not rely on the previously tested 40 scenarios. Overall, we conclude that the proposed benchmark policy is well designed and robust to the proposed MDP model.

Lastly, we demonstrate how we are capable of solving large-scale SAIRPs with our DPHI RL method. For the large-scale SAIRPs, we use the data, parameter settings, and the same set of 40 LHS scenarios as we used for the modest sized instances. However, we note that we scale the demand to be in line with $M' = 100$, that is $\lambda_t' = (^{100}/_{7})\lambda_t$. Furthermore, for the one-month time horizon, $T' = 744$, we use the historical power prices from December 2016, which is the month with the highest power price over the year. Overall, the average computational time is 30018 seconds and we need 126GB of memory to solve these large-scale scenarios using our DPHI RL method. We conclude that DPHI RL is capable of solving large-scale SAIRPs in a reasonable time and using manageable memory.

2.6 Conclusions and Future Research

In this research, we introduced a stochastic scheduling, allocation, and inventory replenishment problem (SAIRP) for a battery swap station wherein there is an interaction between the first-level of inventory (battery charge) and second level of inventory (battery capacity). A main contribution of the SAIRP is considering how battery degradation leads to necessary battery replacement where the degradation is directly caused by the use and charging of batteries. We examine a battery swap station, such as one used for drones or EVs, faced with stochastic demands for swapping. We model the operations at a swap station as a stochastic SAIRP and propose a Markov Decision Process (MDP) model. We define a two-dimensional state space for the MDP representing the number of full batteries and average capacity coupled with a two-dimensional action space representing the number of batteries charged/discharged and number of batteries replaced. We validated the use of average capacity as opposed to each individual battery capacity using a disaggregated MDP model and Monte Carlo simulation. As a result, we believe this research presents an important step forward in modeling battery dependent systems which explicitly incorporates battery capacity degradation. We solve the MDP using backward induction so as to determine the optimal policy indicating the number of batteries to charge, discharge, and replace over time in face of non-stationary charging prices, non-stationary discharging revenue, and capacity-dependent swapping revenue.

We conducted comprehensive computational tests and deduced insights from the results of scenarios generated from Latin Hypercube Sampling. From these results, we analyzed the impact to the expected total reward and optimal policies when the revenue per swap, replacement cost, and the battery degradation factor change. Two key insights that we deduce from these results are as follows. Battery degradation does not have a significant impact on the number of batteries replaced and instead replacement is impacted by the complex interactions between degradation and costs. In most scenarios, the optimal policy indicates to replace all batteries in the swap station at once rather than replacing a portion of the inventory at strategic points in time.

The MDP model for the stochastic SAIRP is large and suffers from the curses of dimensionality. Thus, we propose two approximate solution methods, a heuristic benchmark policy and a double pass with heuristic policy initialization (DHPI) RL method. From the computational tests, we deduce that both methods are competitive and have high performance in terms of optimality gap, satisfied demand gap, and computational time. Generating the benchmark policy requires an insignificant amount of memory and time, thereby making it usable for realistic-sized instances. To design the heuristic benchmark policy, we empirically tested many rules that indicate when and how many batteries are charged and replaced over time. We also leverage the heuristic policy to intelligently initialize our reinforcement learning approach. As a result, we demonstrate that the DPHI RL method can solve large-scale problems in a reasonable time and using a reasonable memory requirements.

Overall, we make steps towards understanding this new SAIRP problem class as we demonstrate that we can solve small problems optimally and other, larger problems using a reinforcement learning and a heuristic benchmark policy. Future work should investigate theoretically proven structure of the optimal policy which dictates when specific actions should be taken. One approximation technique that can be investigated is a policy gradient approach wherein we parameterize the value and policy functions and then find and update the parameter's value such that they are maximizing the average reward per time period. Additionally, researchers should theoretically examine the existence of structure in the optimal policy which would enable the use of more efficient algorithms. Other research should examine different distributions governing the uncertain demand and also others types of uncertainty from the incoming swapped batteries and capacity degradation rate. Lastly, researchers should examine how changing the replacement threshold, θ, changes the optimal policies and possibly the demand for swaps.

Acknowledgement

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.

Bibliography

Abe, M., Nishimura, K., Seki, E., Haruna, H., Hirasawa, T., Ito, S., and Yoshiura, T. (2012). Lifetime prediction for heavy-duty industrial lithium-ion batteries that enables highly reliable system design. *Hitachi Review*, 61(6):259–263.

Abeygunawardane, S. K., Jirutitijaroen, P., and Xu, H. (2013). Adaptive maintenance policies for aging devices using a markov decision process. *IEEE Transactions on Power Systems*, 28(3):3194–3203.

Adler, J. D., Mirchandani, P. B., Xue, G., and Xia, M. (2016). The electric vehicle shortest-walk problem with battery exchanges. *Networks and Spatial Economics*, 16(1):155–173.

Ahiska, S. S., Appaji, S. R., King, R. E., and Warsing, D. P. (2013). A markov decision process-based policy characterization approach for a stochastic inventory control problem with unreliable sourcing. *International Journal of Production Economics*, 144(2):485–496.

AIROBOTICS (2020). Airobotics solution. Last accessed on June 7, 2021 at https://www.airoboticsdrones.com/.

Al-Sabban, W. H., Gonzalez, L. F., and Smith, R. N. (2013). Wind-energy based path planning for Unmanned Aerial Vehicles using Markov Decision Processes. In *2013 IEEE International Conference on Robotics and Automation*, pages 784–789.

Anupindi, R., Morton, T. E., and Pentico, D. (1996). The nonstationary stochastic lead-time inventory problem: Near-myopic bounds, heuristics, and testing. *Management Science*, 42(1):124–129.

Asghari, M. and Al-e-hashem, S. M. J. M. (2020). A green delivery-pickup problem for home hemodialysis machines; sharing economy in distributing scarce resources. *Transportation Research Part E: Logistics and Transportation Review*, 134:101815.

Baek, S. S., Kwon, H., Yoder, J. A., and Pack, D. (2013). Optimal path planning of a target-following fixed-wing UAV using sequential decision processes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2955–2962.

Bookbinder, J. H. and H'ng, B.-T. (1986). Production lot sizing for deterministic rolling schedules. *Journal of Operations Management*, 6(3):349–362.

Chaharsooghi, S. K., Heydari, J., and Zegordi, S. H. (2008). A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45(4):949–959.

Chan, G. and Asgarpoor, S. (2006). Optimum maintenance policy with markov processes. *Electric Power Systems Research*, 76(6):452–456.

Cheevaprawatdomrong, T. and Smith, R. L. (2004). Infinite horizon production scheduling in time-varying systems under stochastic demand. *Operations Research*, 52(1):105–115.

Chen, F., Chen, Z., Dong, H., Yin, Z., Wang, Y., and Liu, J. (2018). Research on the influence of electric vehicle multi-factor charging load on a regional power grid. In *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 163–166.

Cheng, F. and Sethi, S. P. (1999). A periodic review inventory model with demand influenced by promotion decisions. *Management Science*, 45(11):1510–1523.

Cook, R. A. and Lodree, E. J. (2017). Dispatching policies for last-mile distribution with stochastic supply and demand. *Transportation Research Part E: Logistics and Transportation Review*, 106:353–371.

Debnath, U. K., Ahmad, I., and Habibi, D. (2014). Quantifying economic benefits of second life batteries of gridable vehicles in the smart grid. *International Journal of Electrical Power and Energy Systems*, 63:577–587.

DHL Press Release (2016). Successful Trial Integration of DHL Parcelcopter into Logistics Chain. Last accessed on June 7, 2021 at http://www.dhl.com/en/press/releases/releases_2016/all/parcel_ecommerce/successful_trial_integration_dhl_parcelcopter_logistics_chain.html.

DJI (2019). DJI Spreading Wings S1000 specs. Last accessed on November 19, 2020 at https://www.dji.com/spreading-wings-s1000/spec.

Dubarry, M., Truchot, C., Liaw, B. Y., Gering, K., Sazhin, S., Jamison, D., and Michelbacher, C. (2011). Evaluation of commercial lithium-ion cells based on composite positive electrode for plug-in hybrid electric vehicle applications. part II. degradation mechanism under 2C cycle aging. *Journal of Power Sources*, 196(23):10336–10343.

Dunn, B., Kamath, H., and Tarascon, J.-M. (2011). Electrical energy storage for the grid: A battery of choices. *Science*, 334(6058):928–935.

Federgruen, A. and Zipkin, P. (1984). Approximations of dynamic, multilocation production and inventory problems. *Management Science*, 30(1):69–84.

Frederic, G. (2016). LHS DOE generator. Online. Last accessed on June 7, 2021 at https://www.mathworks.com/matlabcentral/fileexchange/57657-lhs-doe-generator.

Fu, D., Zhang, H.-T., Yu, Y., Ionescu, C. M., Aghezzaf, E.-H., and Keyser, R. D. (2019). A distributed model predictive control strategy for the bullwhip reducing inventory management policy. *IEEE Transactions on Industrial Informatics*, 15(2):932–941.

Fu, Y., Yu, X., and Zhang, Y. (2015). Sense and collision avoidance of unmanned aerial vehicles using markov decision process and flatness approach. In *Proceeding of the 2015 IEEE International Conference on Information and Automation*, pages 714–719.

Fusheng, L. (2019). BJEV advocates battery swap service for e-vehicle owners. China Daily. Last accessed on May 23, 2021 at https://www.chinadaily.com.cn/a/201909/09/WS5d75e342a310cf3e3556a816.html.

Gao, Y., Zhao, K., and Wang, C. (2012). Economic dispatch containing wind power and electric vehicle battery swap station. In *IEEE PES Transmission and Distribution Conference and Exposition*, pages 1–7, Orlando, FL.

Gardiner, J. (2017). The rise of electric cars could leave us with a big battery waste problem. Last accessed on June 7, 2021 at The Guardian: https://www.theguardian.com/sustainable-business/2017/aug/10/electric-cars-big-battery-waste-problem-lithium-recycling.

Göransson, L., Karlsson, S., and Johnsson, F. (2010). Integration of plug-in hybrid electric vehicles in a regional. *Energy Policy*, 38(10):5482–5492.

Green, L., Kolesar, P., and Whitt, W. (2007). Coping with time-varying demand when setting staffing requirements for a service system. *Production and Operations Management*, 16:13–39.

Greene, J. H. (1997). *Production and Inventory Control Handbook*. McGraw-Hill., New York, third edition.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692.

Hai, D., Hao, Z., and Ping, L. Y. (2011). Model predictive control for inventory management in supply chain planning. *Procedia Engineering*, 15:1154–1159.

Harris, C. M., Hoffman, K. L., and Saunders, P. B. (1987). Modeling the IRS telephone taxpayer information system. *Operations Research*, 35(4):504–523.

Huang, K. and Liang, Y.-T. (2011). A dynamic programming algorithm based on expected revenue approximation for the network revenue management problem. *Transportation Research Part E: Logistics and Transportation Review*, 47(3):333–341.

Iversen, E. B., Morales, J. M., and Madsen, H. (2014). Optimal charging of an electric vehicle using a markov decision process. *Applied Energy*, 123:1–12.

Jensen, J. (2019). Agricultural drones: How drones are revolutionizing agriculture and how to break into this booming market. UAV Coach. Last accessed on May 21, 2021 at: https://uavcoach.com/agricultural-drones/.

Jiang, C. and Sheng, Z. (2009). Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications*, 36:6520–6526.

Karlin, S. (1960). Dynamic inventory policy with varying stochastic demands. *Management Science*, 6(3):231–258.

Katanyukul, T., Duff, W. S., and Chong, E. K. (2011). Approximate dynamic programming for an inventory problem: Empirical comparison. *Computers & Industrial Engineering*, 60:719–743.

Kim, J., Song, B. D., and Morrison, J. R. (2013). On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems*, 70(1):347–359.

Kim, S. and Moon, I. (2019). Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):42–52.

Kwizera, O. and Nurre, S. G. (2018). Using drones for delivery: A two-level integrated inventory problem with battery degradation and swap stations. In *Proceedings of the Industrial and Systems Engineering Research Conferences*, pages 1–6, Orlando, FL.

Kwon, I.-H., Kim, C. O., Jun, J., and Lee, J. H. (2008). Case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications*, 35(1):389–397.

Lacey, G., Jiang, T., Putrus, G., and Kotter, R. (2013). The effect of cycling on the state of health of the electric vehicle battery. In *48th International Universities' Power Engineering Conference*, pages 1–7, Dublin, Ireland.

Lambert, F. (2018). NIO deploys 18 battery swap stations covering 2,000+ km expressway. electrek. Last accessed on May 23, 2021 at https://electrek.co/2018/11/15/nio-battery-swap-stations-network/.

Lee, C.-Y., Lee, H. L., and Zhang, J. (2015). The impact of slow ocean steaming on delivery reliability and fuel consumption. *Transportation Research Part E: Logistics and Transportation Review*, 76:176–190.

Li, X., Ding, Y., Pan, K., Jiang, D., and Aneja, Y. (2020). Single-path service network design problem with resource constraints. *Transportation Research Part E: Logistics and Transportation Review*, 140:101945.

Liu, J. (2017). Research on electric vehicle fast charging station billing and settlement system. In *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 223–226.

Luo, Z., Qin, H., Zhang, D., and Lim, A. (2016). Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. *Transportation Research Part E: Logistics and Transportation Review*, 85:69–89.

Markoff, J. (2016). Drones marshaled to drop lifesaving supplies over rwandan terrain. New York Times. Last accessed on June 7, 2021 at https://www.nytimes.com/2016/04/05/technology/drones-marshaled-to-drop-lifesaving-supplies-over-rwandan-terrain.html.

McNabb, M. (2017). Asylon charging ahead with battery swap stations. Last accessed on June 7, 2021 at https://dronelife.com/2017/05/15/asylons-charging-ahead-battery-swap-stations/.

Meissner, J. and Senicheva, O. V. (2018). Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems. *European Journal of Operational Research*, 265(1):49–64.

Montgomery, D. (2008). *Design and analysis of experiments*. John Wiley & Sons, New Jersey, 8th edition.

Morrow, K., Karner, D., and Francfort, J. (2008). Plug-in hybrid electric vehicle charging infrastructure review. Technical report, U.S. Department of Energy, Idaho National Laboratory.

Mutzabaugh, B. (2017). Drone taxis? Dubai plans roll out of self-flying pods. Last accessed on May 21, 2021 at https://www.usatoday.com/story/travel/flights/todayinthesky/2017/02/13/dubai-passenger-carrying-drones-could-flying-july/97850596/.

Nair, S. K. (1995). Modeling strategic investment decisions under sequential technological change. *Management Science*, 41(2):282–297.

National Grid (2016). Hourly electric supply charges. Last accessed on June 7, 2021 at https://www.nationalgridus.com/niagaramohawk/business/rates/5_hour_charge.asp.

Nexant, Inc., Air Liquide, Argonne National Laboratory, Chevron Technology Venture, Gas Technology Institute, National Renewable Energy Laboratory, Pacific Northwest National Laboratory, and TIAX LLC (2008). H2A hydrogen delivery infrastructure analysis models and conventional pathway options analysis results. Online. Last accessed on June 7, 2021 at https://www.energy.gov/sites/prod/files/2014/03/f9/nexant_h2a.pdf.

Novas, J. M., Ramello, J. I., and Rodrígueze, M. A. (2020). Generalized disjunctive programming models for the truck loading problem: A case study from the non-alcoholic beverages industry. *Transportation Research Part E: Logistics and Transportation Review*, 140:101971.

Nurre, S. G., Bent, R., Pan, F., and Sharkey, T. C. (2014). Managing operations of plug-in hybrid electric vehicle (PHEV) exchange stations for use with a smart grid. *Energy Policy*, 67:364–377.

Pan, F., Bent, R., Berscheid, A., and Izraelevitz, D. (2010). Locating PHEV exchange stations in V2G. In *First IEEE International Conference on Smart Grid Communications*, pages 173–178, Gaithersburg, MD, USA.

Park, S., Zhang, L., and Chakraborty, S. (2017). Battery assignment and scheduling for drone delivery businesses. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 1–6, Taipei, Taiwan.

Paul Werbos (1989). Backpropagation and neurocontrol: a review and prospectus. In *International 1989 Joint Conference on Neural Networks*, volume 1, pages 209–216.

Peng, M., Liu, L., and Jiang, C. (2012). A review on the economic dispatch and risk management of the large-scale plug-in electric vehicles (PHEVs)-penetrated power systems. *Renewable and Sustainable Energy Reviews*, 16(3):1508–1515.

Pérez Rivera, A. E. and Mes, M. R. (2017). Anticipatory freight selection in intermodal long-haul round-trips. *Transportation Research Part E: Logistics and Transportation Review*, 105:176–194.

Peterson, R. and Silver, E. A. (1979). *Decision systems for inventory management and production planning*. Wiley.

Plett, G. L. (2011). Recursive approximate weighted total least squares estimation of battery cell total capacity. *Journal of Power Sources*, 196(4):2319–2331.

Popper, B. (2016). New Israeli startup, Airobotics, has built a transformer-like base station for its drones. THE VERGE. Last accessed on June 7, 2021 at https://www.theverge.com/2016/6/21/11989734/watch-this-robotic-arm-swap-fresh-batteries-into-an-autonomous-drone.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, New York, NY, USA, 2 edition.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Hoboken, New Jersey, 1st edition.

Rettke, A. J., Robbins, M. J., and Lunday, B. J. (2016). Approximate dynamic programming for the dispatch of military medical evacuation assets. *European Journal of Operational Research*, 254(3):824–839.

Rezvani, Z., Jansson, J., and Bodin, J. (2015). Advances in consumer electric vehicle adoption research: A review and research agenda. *Transportation Research Part D: Transport and Environment*, 34:122–136.

Ribbernick, H., Darcovich, K., and Pincet, F. (2015). Battery life impact of vehicle-to-grid application of electric vehicles. In *28th International Electric Vehicle Symposium and Exhibition*, volume 2, pages 1535–1545, Goyang, Korea. Korean Society of Automotive Engineers.

Romm, J. (2017). Chart of the month: Driven by Tesla, battery prices cut in half since 2014. Last accessed on June 7, 2021 at https://archive.thinkprogress.org/chart-of-the-month-driven-by-tesla-battery-prices-cut-in-half-since-2014-718752a30a42/.

Rong, K. (2012). Research on multi-stage inventory model by markov decision process. *Physics Procedia*, 33:1074–1077.

Roy, B. V., Bertsekas, D., Lee, Y., and Tsitsiklis, J. (1997). A neuro-dynamic programming approach to retailer inventory management. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 4052–4057.

Sathaye, N. and Kelley, S. (2013). An approach for the optimal planning of electric vehicle infrastructure for highway corridors. *Transportation Research Part E: Logistics and Transportation Review*, 59:15–33.

Saxena, S., Floch, C. L., MacDonald, J., and Moura, S. (2015). Quantifying EV battery end-of-life through analysis of travel needs with vehicle powertrain models. *Journal of Power Sources*, 282:265–276.

Scarf, H. (1960). The optimality of (S,s) policies in the dynamic inventory problem. In Arrow, Karlin, and Suppers, editors, *Mathematical Methods in the Social Sciences*, pages 196–202, Stanford, CA. Stanford University Press.

Schneider, F., Thonemann, U. W., and Klabjan, D. (2018). Optimization of battery charging and purchasing at electric vehicle battery swap stations. *Transportation Science*, 52(5):1211–1234.

Schneider, J. J. and Kirkpatrick, S. (2006). *Construction Heuristics*, chapter 7, pages 59–61. In: Stochastic Optimization. Scientific Computation. Springer, Berlin, Heidelberg.

Schroeder, A. and Traber, T. (2012). The economics of fast charging infrastructure for electric vehicles. *Energy Policy*, 43:136–144.

Shavarani, S. M., Nejad, M. G., Rismanchian, F., and Izbirak, G. (2018). Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco. *The International Journal of Advanced Manufacturing Technology*, 95(9):3141–3153.

Shervais, S., Shannon, T. T., and Lendaris, G. G. (2003). Intelligent supply chain management using adaptive critic learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(2):235–244.

Shirk, M. and Wishart, J. (2015). Effects of electric vehicle fast charging on battery life and vehicle performance. In *SAE Technical Paper*, pages 1–13, Detroit, MI. SAE 2015 World Congress & Exhibition, SAE International.

Sioshansi, R. and Denholm, P. (2010). The value of plug-in hybrid electric vehicles as grid resources. *The Energy Journal*, 31(3):1–23.

Soergel, A. (2016). New application for drones: Disaster relief. U.S. News and World Report. Last accessed on June 7, 2021 at https://www.usnews.com/news/articles/2016-06-23/new-application-for-drones-disaster-relief.

Somarin, A. R., Chen, S., Asian, S., and Wang, D. Z. (2017). A heuristic stock allocation rule for repairable service parts. *International Journal of Production Economics*, 184:131–140.

Sun, B., Sun, X., Tsang, D. H., and Whitt, W. (2019). Optimal battery purchasing and charging strategy at electric vehicle battery swap stations. *European Journal of Operational Research*, 279(2):524–539.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2 edition.

Swartzman, G. (1970). The patient arrival process in hospitals: statistical analysis. *Health services research*, 5(4):320–329.

Tan, X., Sun, B., and Tsang, D. H. (2014). Queueing network models for electric vehicle charging station with battery swapping. In *2014 IEEE International Conference on Smart Grid Communications*, pages 1–6.

Tang, H., Xu, L., Sun, J., Chen, Y., and Zhou, L. (2015). Modeling and optimization control of a demand-driven, conveyor-serviced production station. *European Journal of Operational Research*, 243(3):839–851.

Tesla (2017). Supercharger. Last accessed on June 7, 2021 at https://www.tesla.com/supercharger.

Van der Heide, G., Buijs, P., Roodbergen, K., and Vis, I. (2018). Dynamic shipments of inventories in shared warehouse and transportation networks. *Transportation Research Part E: Logistics and Transportation Review*, 118:240–257.

Venkatesan, M. (1984). Production-inventory with equipment replacement-pier. *Operations Research*, 32(6):1286–1295.

Wang, J., Liu, C., Ton, D., Zhou, Y., Kim, J., and Vyas, A. (2011). Impact of plug-in hybrid electric vehicles on power systems with demand response and wind power. *Energy Policy*, 39(7):4016–4021.

Wang, X. and Meng, Q. (2008). Continuous-time dynamic network yield management with demand driven dispatch in the airline industry. *Transportation Research Part E: Logistics and Transportation Review*, 44(6):1052–1073.

Weise, E. (2017). UPS tested launching a drone from a truck for deliveries. USA Today. Last accessed on May 21, 2021 at https://www.usatoday.com/story/tech/news/2017/02/21/ups-delivery-top-of-van-drone-workhorse/98057076/.

Widrick, R. S., Nurre, S. G., and Robbins, M. J. (2018). Optimal policies for the management of an electric vehicle battery swap station. *Transportation Science*, 52(1):59–79.

Williams, R. J. (1988). On the use of backpropagation in associative reinforcement learning. In *IEEE International Conference on Neural Networks*, pages 263–270, San Diego, CA, USA.

Wood, E., Alexander, M., and Bradley, T. H. (2011). Investigation of battery end-of-life conditions for plug-in hybrid electric vehicles. *Journal of Power Sources*, 196(11):5147–5154.

Worley, O. and Klabjan, D. (2011). Optimization of battery charging and purchasing at Electric Vehicle battery swap stations. In *IEEE Vehicle Power and Propulsion Conference*, pages 1–4, Chicago, IL.

Wu, L., Yang, D., Wang, S., and Yuan, Y. (2020). Evacuating offshore working barges from a land reclamation site in storm emergencies. *Transportation Research Part E: Logistics and Transportation Review*, 137:101902.

Xu, B., Oudalov, A., Ulbig, A., Andersson, G., and Kirschen, D. S. (2018). Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140.

Yao, M.-J., Lin, J.-Y., Lin, Y.-L., and Fang, S.-C. (2020). An integrated algorithm for solving multi-customer joint replenishment problem with districting consideration. *Transportation Research Part E: Logistics and Transportation Review*, 138:101896.

Yoon, S. and Albert, L. A. (2020). A dynamic ambulance routing model with multiple response. *Transportation Research Part E: Logistics and Transportation Review*, 133:101807.

Yu, B., Guo, Z., Asian, S., Wang, H., and Chen, G. (2019). Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review*, 125:203–221.

Zhen, L., Ma, C., Wang, K., Xiao, L., and Zhang, W. (2020). Multi-depot multi-trip vehicle routing problem with time windows and release dates. *Transportation Research Part E: Logistics and Transportation Review*, 135:101866.

Zhong, S., Cheng, R., Jiang, Y., Wang, Z., Larsen, A., and Nielsen, O. A. (2020). Risk-averse optimization of disaster relief facility location and vehicle routing under stochastic demand. *Transportation Research Part E: Logistics and Transportation Review*, 141:102015.

Zhou, C., Lee, B. K., and Li, H. (2020a). Integrated optimization on yard crane scheduling and vehicle positioning at container yards. *Transportation Research Part E: Logistics and Transportation Review*, 138:101966.

Zhou, C., Wang, W., and Li, H. (2020b). Container reshuffling considered space allocation problem in container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 136:101869.

3. A Monotone Approximate Dynamic Programming Approach for the Stochastic Scheduling, Allocation, and Inventory Replenishment Problem: Applications to Drone and Electric Vehicle Battery Swap Stations

Amin Asadi Sarah Nurre Pinkley

3.1 Introduction

Electric vehicles (EVs) and drones hold great promise for revolutionizing transportation and supply chains. The United States Department of Energy (2014) reports that EVs can reduce oil dependence and carbon emissions, but vehicle adoption is hindered by range anxiety, purchase price, recharge times, and battery degradation (Rezvani et al., 2015; Saxena et al., 2015). Drone applications have increased in recent years; many organizations are using drones or undergoing testing to use drones for different purposes, including, but not limited to, delivery (CBS NEWS, 2013; Weise, 2017; DHL Press Release, 2016; Wallace, 2015), transportation (Mutzabaugh, 2017), and agriculture (Jensen, 2019). However, drone use is restricted by short flight times, long battery recharge times, and battery degradation (Park et al., 2017; James, 2020; Drones Etc., 2015). An option to overcome these barriers for EVs and drones is a battery swap station. A battery swap station is a physical location that enables the automated or manual exchange of depleted batteries for full batteries in a matter of seconds to a few minutes.

Swap stations have many benefits including their ability to help reduce battery degradation. Battery degradation, or, more specifically, battery capacity degradation, is the act of the battery capacity decreasing over time with use. Each recharge and use of a battery causes a battery to degrade. Degraded battery capacity means EVs and drones have shorter maximum flight times and ranges. Thus, an interesting aspect of managing battery swap stations is that both battery charge and battery capacity are needed; however, the recharging and use of battery charge is the *exact cause* of battery capacity degradation. Thus, this presents a unique problem where recharging batteries, which enables the system to operate in the short-term, is harmful for long-term operation. Although all recharging causes degradation, the regular-rate charging used at swap stations reduces the speed in which batteries degrade as compared to fast-charging (Lacey et al., 2013; Shirk and Wishart, 2015). This increases battery lifespans and causes less environmental waste from disposal. In spite of this benefit, swap stations still must determine when to recharge and replace batteries.

The benefits of battery swap stations are not restricted to decreasing battery degradation. Swap stations also alleviate range anxiety by allowing users to swap their batteries in a couple of minutes. Furthermore, battery swap stations are projected to be a key component within a smart grid through the use of battery-to-grid (B2G) and vehicle-to-grid (V2G). B2G and V2G enable a charged battery to *discharge* the stored energy back to the power grid (Dunn et al., 2011). In practice, several companies such as Toyota Tsusho and Chubu Electric Power in Japan (Nuvve, 2021a,b), NIO and State Grid Corporation in China (Zhang, 2020), and Nissan and E.ON in the UK (Inside EVs, 2020) have installed or plan to install V2G technology. Swap stations can also reduce the purchase price barrier through a business plan where the swap station owns and leases the high-cost batteries (Mak et al., 2013). For the many organizations seeking to use drones, a set of continuously operating drones is often vital. However, continuous operation is difficult because the realized flight time of a drone is often less than the recharge time (James, 2020; Drones Etc., 2015). Thus, automated drone battery swap stations are a promising option because no downtime for recharging is necessary. Given the benefits and applications of swap stations, we examine the problem of optimally managing a battery swap station when considering battery degradation.

We model the operations at a battery swap station using the new class of stochastic scheduling, allocation, and inventory replenishment problems (SAIRPs). In SAIRPs, we decide on the number of batteries to recharge, discharge, and replace over time when faced with time-varying recharging prices, time-varying discharge revenue, uncertain non-stationary demand over time, and capacity-dependent swap revenue. SAIRPs consider the *key interaction* between battery charge and battery capacity and link the use and replenishment (recharging) actions of charge inventory with the degradation and replenishment needs of battery capacity inventory. Battery charge and capacity are linked because each recharge and discharge of a battery causes the battery capacity to degrade, and the level of battery capacity directly limits the maximum amount of battery charge. To replenish battery capacity, we must determine when and how many batteries to replace over time. For SAIRPs, the combination of battery charge and battery capacity is necessary to satisfy non-stationary, stochastic demand over time.

We model the problem as a finite horizon MDP model allowing us to capture the non-stationary elements of battery swap stations over time, including mean battery swap demand, recharging price, and discharging revenue (Puterman, 2005). The MDP's state space is two-dimensional, indicating the total number of fully charged batteries and the average capacity of all batteries at the station. The action of the model is two-dimensional. The first dimension indicates the total number of batteries to recharge or discharge. The second dimension indicates the total number of batteries to replace. The selected action results in an immediate reward, equal to profit, comprised of capacity-dependent revenue from battery swaps, revenue from discharging batteries back to the power grid, cost from recharging batteries, and cost from replacing batteries. The system transitions to a new state according to a discrete probability distribution representing battery swap demand over time, the current state, and the selected action. For our MDP model of the stochastic SAIRP, we seek to determine an optimal policy that maximizes the expected total reward, which is equal to the station's expected total profit. A standard solution method for solving MDPs is backward induction (BI) (Puterman, 2005). We solve a set of modest-sized SAIRPs using BI to provide a baseline for comparing the approximate solution methods; however, as we showed in Chapter 2, BI is not effective for deriving optimal policies for realistic-sized SAIRPs.

Stochastic SAIRPs suffer from the curses of dimensionality, thus, we investigate theoretical properties of the problem to inform more efficient solution methods. We prove that the stochastic SAIRP has a monotone non-decreasing value function in the first, second, and both dimensions of the state. We also prove that general SAIRPs violate the sufficient conditions for the existence of a monotone optimal policy in the second dimension of the state. However, if the number of battery replacements in each decision epoch is constrained to be less than a constant upper bound, we prove there exists a monotone optimal policy for the second dimension of the state in the stochastic SAIRP.

To overcome the curses of dimensionality, we exploit these theoretical results and investigate efficient solution methods. We investigate methods that exploit our proven monotone structure, including Monotone Backward Induction (MBI) (Puterman, 2005) and monotone approximate dynamic programming (ADP) algorithms. First, we examine Jiang and Powell (2015)'s Monotone Approximate Dynamic Programming (MADP) algorithm which exploits the monotonicity of the value function. Next, we propose a new regression-based Monotone ADP algorithm, which we denote MADP-RB. In our MADP-RB, we build upon the foundation of MADP and introduce a regression-based approach to intelligently initialize the value function approximation.

We design a comprehensive set of experiments using Latin hypercube sampling (LHS). We compare the performance of ADP methods with the BI and MBI for the LHS's generated scenarios of a modest size. Experimentally, we show our regression-based ADP generates near-optimal solutions for modest SAIRPs. Besides, using the same LHS scenarios, we solve large-scale SAIRPs with our proposed ADP algorithms. We demonstrate that our proposed ADP approaches can overcome the inherent curses of dimensionality of SAIRPs that BI, and MBI failed to succeed.

Main Contributions. The main contributions of this work are as follows: (i) we demonstrate that stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state and prove the existence of a monotone optimal policy for the second dimension of state when an upper bound is placed on the number of batteries replaced in each decision epoch; (ii) we prove the monotone structure for the MDP value function; (iii) we propose a regression-based monotone ADP method by utilizing the theoretical structure of the MDP optimal value function to intelligently approximate the initial value function and make updates in each iteration; (iv) we computationally demonstrate the superior performance of our regression-based monotone ADP algorithm and deduce managerial insights about managing battery swap stations.

The remainder of the chapter is organized as follows. In Section 3.2, we outline literature relevant to our modeling approach, solution approaches, and EV and drone applications. In Section 3.3, we formally define the stochastic scheduling, allocation, and inventory replenishment problem as a two-dimensional Markov Decision Process. In Section 3.4, we present theoretical results for the stochastic SAIRP. In Section 3.5, we present solution methods and outline the monotone ADP algorithm with regression-based initialization to solve stochastic SAIRP instances. In Section 3.6, we present results and insights from computational tests of the solution methods and realistic instances of the stochastic SAIRP. We summarize the contributions in Section 3.7 and provide opportunities for future work.

3.2 Literature Review

There is growing interest surrounding electric vehicles (EVs) and drones in industry and academia. We proceed by discussing the relevant literature pertaining to (i) the EV and drone swap station application; (ii) the background knowledge for the proposed approach using aspects of optimal timing and reliability, inventory management, and equipment replacement problems; (iii) the scientific works that explain the lithium-ion battery degradation process; and (iv) ADP approaches that address the curses of dimensionality. To the best of our knowledge, no past research has derived the structure of the optimal policy and value function for the scheduling, allocation, and inventory replenishment problem nor solved the realistic-sized instances of SAIRPs to derive insights for managing the operations at a battery swap station faced with battery degradation.

Swap stations were initially introduced for EVs and thus have a more extensive research base. However, there is growing interest surrounding drone battery swap stations. We first examine the work on managing the internal operations of a battery swap station that are most similar to the model presented in this chapter and Chapter 2. Widrick et al. (2018) develop an inventory control MDP for a swap station that only considers the number of batteries to recharge and discharge over time but excludes battery capacity levels, degradation, and replacement. They prove the existence of a monotone optimal policy only when the demand is governed by a non-increasing discrete distribution (e.g., geometric). Nurre et al. (2014) also consider determining the optimal charging, discharging, and swapping at a swap station using a deterministic integer program that excludes uncertainty. Worley and Klabjan (2011) examine an EV swap station with uncertainty and seek to determine the number of batteries to purchase and recharge over time. Note, purchasing batteries is fundamentally different from battery replacement in SAIRPs. Worley and Klabjan (2011) examine the one-time purchase of batteries to open a swap station and do not consider purchasing decisions over time. Contrarily, we assume the initial number of batteries at the swap station is previously determined and instead consider replacing batteries over time. Sun et al. (2014) propose a constrained MDP model for determining an optimal charging policy at a single battery swap station and examine the tradeoffs between quality of service for customers and energy consumption costs.

Other research considers a mix of long-term strategic and short-term operational swap station decisions. Schneider et al. (2018) consider a network of swap stations that seeks to determine the long-term number of charging bays and batteries to locate at each station and the short-term number of batteries to recharge over time. Schneider et al. (2018) do consider charging *capacity*; however, their use of capacity indicates the number of batteries that can be recharged at one time in the station and do not model *battery capacity*. Kang et al. (2016) propose the EV charging management problem, which determines the optimal locations for a network of swap stations and further determines the charging policy for each location. Their definition of charging policy only considers charging and excludes discharging or replacement. Excluding the explicit charging actions over time, Zhang et al. (2014) determine the number of batteries that are necessary for swapping over time. For further studies in the area of EV operations management, we refer the reader to a review by Shen et al. (2019).

A common limitation of the aforementioned research is that it fails to account for battery degradation. To the best of our knowledge, there are very few articles that consider battery degradation. Chapter 2 research is the first to introduce stochastic SAIRPs for managing battery swap stations with degradation. However, in Chapter 2, we do not theoretically analyze this problem class, do not introduce intelligent approximate dynamic solution methods that exploit the theoretical results, and do not provide insights from solving realistic-sized SAIRPs.

Others have examined battery degradation in a deterministic setting without any uncertainty (Kwizera and Nurre, 2018; Park et al., 2017; Tan et al., 2019). Sarker et al. (2015) consider the problem of determining the next day operation plan for a battery swap station under uncertainty. They do consider battery degradation; however, they solely penalize battery degradation with a cost in the objective and do not link it to a reduction in operational capabilities.

Others have examined battery swap stations from different perspectives. Researchers have examined how to find the optimal number and location of swap stations in a system (Shavarani et al., 2018; Kim et al., 2013; Hong et al., 2018). Extending this idea further, Yang and Sun (2015) look to locate swap stations and route vehicles through the swap stations. Others have examined how to locate and/or operate swap stations that are coordinated with green power resources (Pan et al., 2010), stabilize uncertainties from wind power (Gao et al., 2012), or coordinate with the power grid (Dai et al., 2014).

Our research is related to optimal timing and reliability problems. There is a rich literature on finding the optimal timing of decisions to maximize systems' lifespan and reliability. For instance, researchers maximize the expected quality-adjusted life years by finding the optimal timing of living-donor liver transplantation (Alagoz et al., 2004), biopsy test (Chhatwal et al., 2010; Zhang et al., 2012), and replacement of an Implantable Cardioverter Defibrillator generator (Khojandi et al., 2014). There are two options for the actions in these works (e.g., transplant/wait, take/skip the biopsy test, replace/not replace). However, our action determines the *number* of batteries to recharge/discharge and replace in each epoch because it is not a single battery that enables the station to operate. Instead, it a set of batteries that enables operation, which creates a significantly larger action space that is dependent on the number of batteries at the station. Similar to our work is that of (Bloch-Mercier, 2001), which determines the optimal timing and duration of a degrading repairable system. There is extensive research in the nexus of optimization, reliability, and systems maintenance. We refer the reader to the recent review paper by de Jonge and Scarf (2020) for further study.

Our research can be placed under the umbrella of inventory management and equipment replacement problems with stochastic elements. There is a large research base examining these types of problems under different characteristics. We proceed by reviewing a small sample of this body of knowledge by focusing on foundational work and research most similar to the scope of this research. Researchers have extensively studied inventory problems, including those with stochastic demand (Porteus, 2002), two- and multi-echelon supply chains (Clark and Scarf, 1960; Clark, 1972), and multiple products (DeCroix and Arreola-Risa, 1998). A desirable feature of the solutions to inventory problems is that the optimal policy has a simple structure. A classic example of such an optimal policy is the (s,S) policy that indicates to order up to S units when the inventory level drops below s (Scarf, 1960). Others have examined more sophisticated inventory problems which include scheduling production (Laan and Salomon, 1997; ElHafsi, 2009; Maity, 2011; Golari et al., 2017), performing maintenance or replacement (Horenbeek et al., 2013), and ordering spare parts for maintenance (Elwany and Gebraeel, 2008; Rausch and Liao, 2010). Additionally, researchers have examined perishable inventory that degrades over time (Nahmias, 1982) or inventory that can be recycled or remanufactured in a closed-loop supply chain (Toktay et al., 2000; Zhou et al., 2011; Govindan et al., 2015).

The proposed work is distinct from this previous literature as it links the actions of recharging batteries to the actions that must be taken for replacing battery capacity. No prior work includes the counter-intuitive property that the act of maintaining the system in the short term (e.g., through recharging batteries which can be analogous to short-term maintenance or short-term inventory replenishment) is harmful for long-term performance (e.g., future need to replace equipment or replenish other types of inventory).

A novel component of our work is the consideration of battery degradation within the decision-making process. Battery degradation is most traditionally measured based on calendar life or cycles, where a cycle consists of one use and one recharge (Lacey et al., 2013; Ribbernick et al., 2015). Using physical experiments, simulation, and mathematical modeling, researchers aim to capture the rate of battery degradation for different batteries and conditions such as temperature and depth of discharge (Plett, 2011; Abe et al., 2012; Ribbernick et al., 2015; Hussein, 2015; Dubarry et al., 2011). We approximate battery degradation using a linear degradation factor derived from the work of Lacey et al. (2013) and Ribbernick et al. (2015), as is consistent with other research using a linear forecast (Xu et al., 2018; Abdollahi et al., 2015; Wood et al., 2011).

Our MDP model suffers from the curses of dimensionality due to the very large size of all MDP elements together, including state and action spaces, transition probability, and reward. Approximate dynamic programming (ADP) is a method that has had great success in determining near-optimal policies for large-scale MDPs (Powell, 2011). Researchers have used ADP methods to solve problems in energy, healthcare, transportation, resource allocation, and inventory management (Bertsimas and Demir, 2002; Powell and Topaloglu, 2005; Simao et al., 2009; Erdelyi and Topaloglu, 2010; Maxwell et al., 2010; Çimen and Kirkbride, 2013; Meissner and Senicheva, 2018; Çimen and Kirkbride, 2017; Nasrollahzadeh et al., 2018). Jiang and Powell (2015) propose a monotone ADP algorithm that is specifically designed for problems with monotone value functions. In this research, we prove that the value function of the stochastic SAIRP has a nondecreasing monotone structure. Hence, we utilize Jiang and Powell (2015)'s monotone ADP algorithm and enhance it by adding a regression-based initialization.

3.3 Problem Statement

In this section, we present and model the Markov Decision Process (MDP) model of the scheduling, allocation, and inventory replenishment problem (SAIRP) that considers stochastic demand for swaps over time, non-stationary costs for recharging depleted batteries, non-stationary revenue from discharging, and capacity-dependent swap revenue. The MDP model captures the dynamic average battery capacity over time, the associated replacement policies, and the interaction between battery charge and battery capacity at a battery swap station. We note, this model was originally presented in Chapter 2; however, we believe it is necessary to provide the reader with the formal problem definition to enable understanding of the main theoretical and algorithmic contributions that follow. We use a finite horizon MDP to capture the high variability of data over time, including the mean demand for battery swaps, the price for recharging batteries, and the revenue earned from discharging batteries back to the power grid. The uncertainty in the system is the stochastic demand for battery swaps (i.e., exchange of a depleted battery for a fully-charged battery). We model this uncertainty (stochastic demand) using the random variable, D_t, for each time period t. These random variables are explicitly used to calculate the transition probabilities. The objective is to maximize the expected total reward of the swap station and determine optimal policies which dictate how many batteries to recharge, discharge, and replace over time. For our model, the expected total reward equals the expected total profit calculated as the revenue from satisfying demand and discharging batteries to the power grid minus the costs from recharging and replacing batteries.

We formulate our MDP model with the following elements. We define T as the finite set of decision epochs, which are the discrete periods in time in which decisions are made. By defining N as the terminal epoch, $T = \{1,\ldots,N-1\}$, $N < \infty$.

We denote the two-dimensional state of the system at time t, $s_t = (s_t^1, s_t^2) \in S = (S^1 \times S^2)$, as the total number of fully charged batteries, $s_t^1 \in S^1$, and the average capacity of all batteries, $s_t^2 \in S^2$, at the swap station. In the design of s_t^1, we only consider that batteries are either fully charged or depleted. The number of full batteries at time t, s_t^1, is an integral value between 0 and M, where M is the total number of batteries in the station, thus, $S^1 = \{0,1,2,\ldots,M\}$.

We use an aggregated MDP in which we track the discretized average battery capacity rather than a disaggregated MDP, which tracks each battery capacity individually, to reduce the curses of dimensionality from the second dimension of the state. The disaggregated MDP severely suffers from the curse of dimensionality as the state space's size grows exponentially as the number of batteries increases. We discretize the average battery capacity where $S^2 = \{0, \theta, \theta+\varepsilon, \theta+$

$2\varepsilon, \ldots, 1\}$, in which θ equals the lowest acceptable average battery capacity and ε in the discretized capacity increment. State zero in S^2 is an absorbing state representing that the average battery capacity dropped below θ. To discourage the station from allowing the battery capacity to drop below θ thereby resulting in lower quality batteries at the station, we disallow charging, discharging, swapping, and replacement when in this absorbing state. Hence, the set of feasible actions when in an absorbing state, $s_t^2 < \theta$ or $s_t^2 = 0$, only includes no recharge/discharge and no replacement. We note, with this aggregated modeling proposed in Chapter 2, the problem size and complexity are reduced, which is not always necessary when using approximate solution methods. However, the aggregated model allows us to benchmark the performance of new and existing approximate solution methods and analyze larger SAIRP instances. Further, in Chapter 2, we previously showed that the results do not significantly change with aggregation.

We denote the two-dimensional action to represent the number of batteries to recharge/discharge, a_t^1, and the number of batteries to replace, a_t^2, at time t. In our aggregated MDP model, there is no known difference between the capacity of batteries as we only track the average capacity of all batteries. In reality, swap stations, applying the aggregated MDP, may track/not track the capacity of each battery. If, consistent with the model, the swap station does not track individual battery capacity values, we assume the specific batteries that are selected to be recharged/replaced or discharged/swapped are *arbitrarily* selected from the set of empty and fully-charged batteries, respectively. However, if the swap station does track the individual battery capacity value, we assume that the station selects to recharge/discharge and swap batteries with the highest capacity values and selects to replace batteries with the lowest capacity values. With this selection mechanism, individual battery capacity values will be closer to the average battery capacity of the system and, thus, further emphasizes the aggregated modeling decision. Regarding the first dimension of the action, a_t^1, we attribute a positive value to the number of batteries to recharge, a_t^{1+}, and a negative value to the number of batteries to discharge, a_t^{1-}. To clarify the distinction between recharging and discharging actions, we define positive recharging, a_t^{1+}, and discharging actions, a_t^{1-}, with Equations (3.1) and (3.2). We note that only dealing with the positive number of batteries that are recharged or discharged using Equations (3.1) and (3.2) is helpful to clarify the forthcoming state transition, probability transitions, and reward calculations.

$$a_t^{1+} = \begin{cases} a_t^1 & \text{if } \ a_t^1 \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

$$a_t^{1-} = \begin{cases} |a_t^1| & \text{if } \ a_t^1 < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The action a_t^1 represents both the number of batteries that are recharged, when a_t^1 is positive, and the number of batteries that are discharged, when a_t^1 is negative. We designed the action in this way as it is not beneficial to recharge and discharge at the same epoch, as they will cancel each other out and cause the capacity to degrade. Thus, we select one value for a_t^1 for each time t, and state, s_t. Depending on whether the selected action is positive or negative indicates whether recharging or discharging will occur. We denote the number of plug-ins in the station as Φ. We assume all plug-ins are capable of supplying energy from the grid to recharge batteries and receiving energy from batteries discharged using Battery to Grid (Dunn et al., 2011). We define the first dimension of action as $a_t^1 \in A_t^1 = \{\max(-s_t^1, -\Phi), \ldots, 0, \ldots, \min(M - s_t^1 - a_t^2, \Phi)\}$, which limits the number of discharged batteries by the minimum of the number of plug-ins and the number of full batteries ($-\min(s_t^1, \Phi) = \max(-s_t^1, -\Phi)$) and limits the number of recharged batteries by the minimum of the number of plug-ins and the number of depleted batteries that were not replaced. In the second dimension of the action space, $a_t^2 \in A_t^2 = \{0, \ldots, M - s_t^1\}$, we only allow depleted batteries to be replaced at each epoch t which arrive in epoch $t+1$ with full charge and capacity. We define $A_{s_t} = (A_{s_t}^1 \times A_{s_t}^2) \subseteq (A_t^1 \times A_t^2)$ as the set of feasible actions for the state s_t at time t. In our model, the set of feasible actions when in an absorbing state, $s_t^2 < \theta$ or $s_t^2 = 0$, only includes no recharge/discharge and no replacement; i.e., $A_{(s_t^1, 0)} = \{(0,0)\}$.

In Figure 3.1, we display the timing of the operations at the swap station including recharging, discharging, replacing, and swapping between epochs t and $t+1$. We assume that the time between two consecutive epochs is sufficient to recharge or discharge a battery completely. In our model, we could preemptively recharge, discharge or replace batteries for future time periods. Therefore, depleted (full) batteries selected for recharging (discharging) in epoch t are fully charged (depleted) at the start of epoch $t+1$. When stochastic demand for a battery swap arrives in epoch t, we can swap up to the number of fully-charged batteries in our inventory which equals the number of fully-charged batteries at the start of t minus the number of discharged batteries. We subtract the fully-charged batteries assigned to be discharged as they are unavailable for swapping until the next decision epoch.

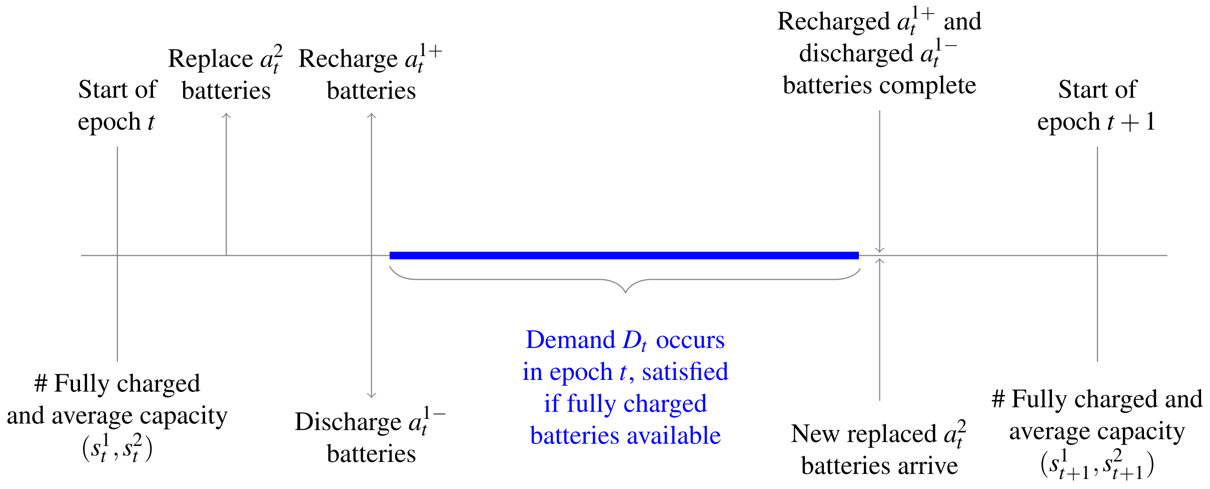

Start of epoch t
Replace a_t^2 batteries
Recharge a_t^{1+} batteries
Recharged a_t^{1+} and discharged a_t^{1-} batteries complete
Start of epoch $t+1$
# Fully charged and average capacity (s_t^1, s_t^2)
Discharge a_t^{1-} batteries
Demand D_t occurs in epoch t, satisfied if fully charged batteries available
New replaced a_t^2 batteries arrive
# Fully charged and average capacity (s_{t+1}^1, s_{t+1}^2)

Figure 3.1: Diagram outlining the timing of events for the SAIRP model.

Transition probabilities indicate the likelihood of transitioning between states when considering the uncertainty of the system. In our MDP model, the uncertainty in the system is the stochastic demand for battery swaps (i.e., exchange of a depleted battery for a fully-charged battery) at each decision epoch t, D_t. The amount of satisfied swap demand in epoch t equals $\min\{D_t, s_t^1 - a_t^{1-}\}$ wherein the second term indicates the number of full batteries that are not already discharging at t. We outline the state transition for the first dimension of the state in Equation (3.3) which determines the number of full batteries in epoch $t+1$ based on the number of full, recharged, discharged, replaced, and swapped batteries in epoch t.

$$s^1_{t+1} = s^1_t + a^2_t + a^{1+}_t - a^{1-}_t - \min\{D_t, s^1_t - a^{1-}_t\}. \tag{3.3}$$

The second state transitions according to Equation (3.4), which determines the future average capacity in $t+1$ based on the current average capacity and the number of full, recharged, discharged, and replaced batteries in epoch t. We assume that all batteries swapped at time t have a capacity equal to the average capacity of the batteries at the swap station. We justify the assumption with the following logic. Batteries previously swapped in epoch $t_1 < t$, which are in use outside of the station between t_1 and t and need to be swapped again in epoch t, have a capacity similar to the average station capacity at t when the swap station is used regularly (i.e., $t - t_1$ is small). We define δ^C to represent the amount of battery capacity degradation from one battery cycle. We adopt the cycle-based degradation measure (Abdollahi et al., 2015; Lacey et al., 2013) and assume that batteries do not degrade when not in use. Further, without loss of generality, we attribute the degradation from a full cycle to the recharge/discharge portion of the cycle. We use *round*() to represent that Equation (3.4) returns values in the discretized state space, S^2, with ε precision.

$$g^2(s^1_t, s^2_t, a^1_t, a^2_t) = s^2_{t+1} = round\left(\frac{(s^2_t - \delta^C)(a^{1+}_t + a^{1-}_t) + a^2_t + s^2_t(M - a^{1+}_t - a^{1-}_t - a^2_t)}{M}\right). \tag{3.4}$$

In the first term in the numerator of Equation (3.4), we multiply the summation of the number of recharged ($a^{1+}_t > 0$) and discharged ($a^{1-}_t > 0$) batteries by the reduced average capacity ($s^2_t - \delta^C$) due to the recharging/discharging actions. The second term adds the a^2_t replaced batteries with 100% capacity. The third term maintains the same capacity for batteries not recharged, discharged, and replaced. These terms are all averaged over the M batteries in the swap station. The system enters the absorbing state $0 \in S^2$ when the average capacity is less than θ. To discourage entrance into this absorbing state, no recharging, discharging, swapping, or replacement is allowed. This setting ensures that swap stations should take appropriate actions *before* allowing the average capacity to drop below θ. Thus, the transition of the second dimension of the state is precisely defined with Equation (3.5).

$$f^2(s_t^1, s_t^2, a_t^1, a_t^2) = s_{t+1}^2 = \begin{cases} g^2(s_t^1, s_t^2, a_t^1, a_t^2) & \text{if} \quad g^2(s_t^1, s_t^2, a_t^1, a_t^2) \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

In Equation (3.6), we define the probability of transitioning from state $s_t = (s_t^1, s_t^2)$ in epoch t to the state $j = (j^1, j^2)$ in epoch $t+1$ when action $a_t = (a_t^1, a_t^2)$ is taken.

$$p(j^1, j^2 \mid s_t^1, s_t^2, a_t^{1+}, a_t^{1-}, a_t^2) = \begin{cases} p_{s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1} & \text{if } a_t^2 + a_t^{1+} < j^1 \leq s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} \text{ and} \\ & j^2 = f^2(s_t^1, s_t^2, a_t^1, a_t^2), \\ q_{s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1} & \text{if } j^1 = a_t^2 + a_t^{1+} \text{ and } j^2 = f^2(s_t^1, s_t^2, a_t^1, a_t^2), \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

We define $p_j = P(D_t = j)$ and $q_u = \sum_{j=u}^{\infty} p_j = P(D_t \geq u)$. Each probability in Equation (3.6) depends on the number of batteries swapped, i.e., $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1$ (see Equation (3.3)). When no batteries are swapped in epoch t, the station still has $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-}$ fully charged batteries at epoch $t+1$. Instead, if all available fully-charged batteries in epoch t are swapped, the station will have $a_t^2 + a_t^{1+}$ fully charged batteries at epoch $t+1$, which are the result of the a_t^2 replaced and a_t^{1+} recharged batteries in epoch t.

In Equation (3.6), the probability of transitioning to another state is non-zero only when Equation (3.5) is satisfied. When the transition probability is $P(D_t = s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1)$, the demand for swaps is less than or equal to the number of full batteries available for swapping (as in condition 1 of Equation (3.6)). Alternatively, when the demand for swaps is greater than the number of available full batteries, the state transitions according to the cumulative probability $P(D_t \geq s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1)$. If Equation (3.5) is not satisfied, j^1 is lower than the total number of batteries recharged and replaced ($a_t^2 + a_t^{1+}$), or j^1 exceeds than the maximum number of fully charged batteries, $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-}$, the probability of transition is zero.

To clarify the transition probability function, we illustrate using an example. Consider the case when at epoch t, the swap station has 80 full-batteries and 20 depleted batteries in inventory (i.e., $M = 100, s_t^1 = 80$), the average battery capacity equals 0.85, and we take the action to recharge 10 batteries (i.e., $a_t^1 = a_t^{1+} = 10$) and replace 5 batteries (i.e., $a_t^2 = 5$). For this example, we assume recharging or discharging for one time period results in a capacity degradation equal to 0.01 (i.e., $\delta^C = 0.01$) and the discretized capacity increment is also 0.01 (i.e., $\varepsilon = 0.01$). If there is no demand for battery swaps (i.e., $D_t = 0$), at epoch $t+1$ the station will have 95 full batteries with a discretized average capacity equal to 0.86. Thus, the probability of transitioning to a state with more than 95 full batteries or an average capacity not equal to 0.86 is zero. Contrarily, if the demand for swaps is 80 or more (i.e., $D_t \geq 80$), then all full batteries in inventory will be swapped and the number of full batteries at at epoch $t+1$ equals $a_t^2 + a_t^{1+} = 10+5 = 15$. Thus, the probability of transitioning to a state with less than 15 full batteries is zero. Further, the probability of transitioning to a state with exactly 15 full batteries and average capacity equal to 0.86 indicates that demand for swaps met or exceeded $s_t^1 - a_t^{1-} = 80$. Lastly, consider the case that we transition to a state with 30 full batteries and average capacity equal to 0.86. The 30 full batteries is between the minimum, $a_t^2 + a_t^{1+} = 15$, and maximum, $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} = 95$ number of full batteries; thus, we know that $a_t^2 + a_t^{1+} = 15$ batteries arrive at the end of t indicating that we swapped $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - j^1 = 80+5+10-0-30 = 65$ batteries at epoch t. As follows, the probability of transitioning to this state equals the probably that demand for swaps equals 65, i.e., $P(D_t = 65)$.

The actions taken seek to maximize the expected total reward. The expected total reward depends on the immediate reward earned at each epoch. Specifically, the immediate reward is the profit earned. In our setting, swap stations earn revenue from swapping and/or discharging fully-charged batteries and incur costs to recharge and/or replace depleted batteries. We calculate the immediate reward at epoch t according to the state of the system $s_t = (s_t^1, s_t^2)$, the taken action $a_t = (a_t^1, a_t^2)$, and the future state $s_{t+1} = (s_{t+1}^1, s_{t+1}^2)$. Specifically, the immediate reward is calculated according to Equation (3.7),

$$r_t(s_t, a_t, s_{t+1}) = \rho_{s_t^2}(s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - s_{t+1}^1) - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2, \tag{3.7}$$

where $s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - s_{t+1}^1$ equals the number of batteries swapped and the time-dependent recharging cost, discharging revenue, and replacement cost are defined as K_t, J_t, and L_t, respectively. We note that SAIRPs consider two aspects of a battery, charge and capacity. In this model, the fully-charged/empty batteries are not necessarily full-capacity as they might already be degraded due to the previous recharge/discharge actions. Thus, the average capacity of batteries can take a value less than 100%. We assume the realized swap revenue depends on the current average capacity. Thus, we define $\rho_{s_t^2}$ to be the capacity-dependent revenue per battery swapped in Equation (3.8).

$$\rho_{s_t^2} = \beta\left(1 + \frac{s_t^2 - \theta}{1 - \theta}\right) = \frac{\beta(1 + s_t^2 - 2\theta)}{1 - \theta}. \tag{3.8}$$

We set $\beta \geq \max_{t \in T} J_t$ to ensure the swap station is profitable with each battery swapped (i.e., the swap revenue is no less than the maximum recharging cost). We use the average capacity of batteries as the indicator of the quality of batteries in the station when developing the revenue per swap function. Revenue per battery swap is a linear function of the average capacity of batteries in the station. This setting ensures that the stations can gain higher revenue when the average capacity is higher. It also provides an incentive for swap stations to replace batteries for higher revenue and benefits customers by receiving higher quality batteries. In the design of Equation (3.8), when the average capacity is at the lowest operational value ($s_t^2 = \theta$), the revenue per swap $\rho_{s_t^2}$ equals β, which is at least equal to the maximum price paid for recharging batteries. When the swap station has an average battery capacity equal to 1, $s_t^2 = 1$, then $\rho_{s_t^2} = 2\beta$ which equates to a higher revenue earned due to higher customer satisfaction from swapping a higher quality battery. Hence, in our design, the revenue per swap has a value between $[\beta, 2\beta]$ depending on the average capacity of batteries in the station at time t. We calculate the terminal reward in Equation (3.9) as the potential revenue from swapping all remaining fully charged batteries provided the average battery capacity is at least θ.

$$r_N(s_N) = \begin{cases} \rho_{s_N^2} s_N^1 & \text{if } s_N^2 \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Using the probability transition function and the immediate reward, we define the immediate expected reward in Equation (3.10).

$$r_t(s_t, a_t) = \sum_{s_{t+1} \in S} \left[p_t(s_{t+1} \mid s_t, a_t)(\rho_{s_t^2}(s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - s_{t+1}^1)) \right] - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2. \tag{3.10}$$

We define the decision rules, $d_t(s_t) : s_t \rightarrow A_{s_t}$, as a function of the current state and time. Our decision rules determine the selected action $a_t \in A_{s_t}$ when the system is in s_t at decision epoch $t \in T$. In our problem setting, we use deterministic Markovian decision rules because we choose which action to take provided we know the current state (Puterman, 2005). A policy π consists of a sequence of decision rules $(d_1^\pi(s_1), d_2^\pi(s_2), \ldots, d_{N-1}^\pi(s_{N-1}))$ for all decision epochs. The expected total reward of policy π, denoted $\upsilon_N^\pi(s_1)$ when the system starts in state s_1 at time t=1 is calculated according to Equation (3.11).

$$\upsilon_N^\pi(s_1) = \mathbb{E}_{s_1}^\pi \left[\sum_{t=1}^{N-1} r_t(s_t, a_t) + r_N(s_N) \right]. \tag{3.11}$$

In Section 3.5, we describe our solution methodology to find optimal/near-optimal solutions to maximize the expected total reward of the stochastic SAIRPs.

3.4 Theoretical Results

In this section, we prove theoretical properties regarding the structure of the optimal SAIRP policy and value function. First, we show that the stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state. Second, we prove the existence of a monotone optimal policy for the second dimension of the state in a special case of the SAIRP. Lastly, we prove the monotonicity of the value function when considering the first, second, and both dimensions of the state. In the remainder of this section, we present the main theorems and point the reader to the appendices for the formal mathematical proofs.

3.4.1 Monotone Policy

Our investigation in proving the structure of an optimal policy for the SAIRP is motivated by the desire to exploit efficient algorithms that require less computational effort to find optimal policies and increase the ability to solve larger problem instances (Puterman, 2005). Widrick et al. (2018) examined the problem of managing a battery swap station when only considering battery charge, or equivalently the first dimension of our MDP model. They proved the existence of a monotone optimal policy when demand is governed by a non-increasing discrete distribution. In our investigation of the second dimension, our intuition was that monotonicity would be preserved for the stochastic SAIRP. Informally, this equates to the optimal policy indicating to replace more batteries when the average capacity is lower. However, in Lemma 1, we prove a counter-intuitive result that, in general, the sufficient conditions for the optimality of a monotone policy in the second dimension of the state do not exist for the stochastic SAIRPs. Instead, we are able to prove the existence of a monotone optimal policy for the second dimension of the state when an upper bound is placed on the number of batteries replaced at the swap station in each decision epoch.

First, we formally define a monotone optimal policy. A non-increasing monotone policy π has the property that for any $s_i, s_j \in S$ with $s_i \leq s_j$ (for multi-dimensional states, please see the partial ordering definition in Definition 2 of Appendix 3.B), there exist decision rules $d_t^{\pi}(s_i) \geq d_t^{\pi}(s_j)$ for each $t = 1, \dots, N-1$ (Puterman, 2005). The sufficient conditions for the existence of a monotone optimal policy in the second dimension of the state are as follows (Puterman, 2005).

1. $r_t(s_t^2, a_t^2)$ is non-decreasing in s_t^2 for all $a_t^2 \in A'$.
2. $q_t(k \mid s_t^2, a_t^2)$ is non-decreasing in s_t^2 for all $k \in S^2$ and $a \in A'$.
3. $r_t(s_t^2, a_t^2)$ is a subadditive function on $S^2 \times A'$.
4. $q_t(k \mid s_t^2, a_t^2)$ is subadditive on $S^2 \times A'$ for every $k \in S^2$.

5. $r_N(s_N)$ is non-decreasing in s_N^2.

Where A' includes all possible actions for the second dimension of the action space. Specifically, $A' = \{\cup_{s_t \in S} A_{s_t}^2\}$. We note that $q_t(k \mid s,a) = \sum_{j=k}^{\infty} p_t(j \mid s,a)$, which is the sum of the probabilities from k to ∞, in general. For the second dimension, we have $q_t(k \mid s_t^2, a_t^2) = \sum_{j^2=k}^{\infty} p_t(j^2 \mid s_t^2, a_t^2)$.

In Lemma 1, we prove that one of the aforementioned conditions is not satisfied for stochastic SAIRPs. In Theorem 1, we are able to prove that a monotone optimal policy in the second dimension of the state does exist when there is an upper bound on the number of batteries replaced in each decision epoch. We refer the reader to Appendix 3.A for full details of the proof of Lemma 1 and Theorem 1.

Lemma 1. *The stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state.*

Theorem 1. *There exist optimal decision rules $d_t^* : S \to A_{s_t}$ for the stochastic SAIRP which are monotone non-increasing in the second dimension of the state for $t = 1,\ldots,N-1$ if there is an upper-bound U on the number of batteries replaced at each decision epoch where $U = \frac{M\varepsilon}{2(1-s_t^2)}$, when M is the number of batteries at the swap station and ε is the discretized increment in capacity.*

We provide an example to explain the optimality of the monotone policy in the second dimension of the state. Consider a swap station with $M = 100$ batteries, a discretized capacity increment $\varepsilon = 0.01$, and a replacement threshold $\theta = 0.8$. The monotone policy is optimal when the maximum number of batteries replaced per epoch, U, is between 2 and 50. The specific value between 2 and 50 depends on the value of the average capacity. If the average capacity $s_t^2 = 0.8$, then $U = \frac{100(0.01)}{2(1-0.8)} = 2$ whereas if $s_t^2 = 0.99$, then $U = \frac{100(0.01)}{2(1-0.99)} = 50$. We note that when $s_t^2 = 1$, then $U = \frac{100(0.01)}{2(1-1)} = \infty$ meaning there is no limit on the number of replaced batteries. However, as the average capacity is already at the highest value of 1, it is not advantageous for swap stations to incur the cost for replacing a full capacity battery after which the average capacity will remain at 1. Although there is a restriction on the number replaced in each epoch, there are no restrictions on the consistent replacement of batteries over multiple consecutive decision epochs.

3.4.2 Monotone Value Function

We now investigate the structure of the value function for stochastic SAIRPs. Although proving that a value function has a monotone structure is a weaker result than proving the structure of an optimal policy, it enables the application of computationally efficient solution methods. We prove that the MDP value function for the stochastic SAIRP is monotone non-decreasing in the first, second, and both dimensions. These results directly motivate our selection of efficient approximate dynamic programming algorithms.

A value function $V(s)$ is monotone non-decreasing in state s, if for any $s_i, s_j \in S$ with $s_i \leq s_j$ we have $V(s_i) \leq V(s_j)$ for any given action in any decision epoch t (Papadakia and Powell, 2007; Jiang and Powell, 2015). The MDP value function for the stochastic SAIRP is given in Equation (3.12) which is comprised of the immediate expected reward (as given by Equation (3.7)) and the transition probabilities (as given by Equation (3.6)). In Theorem 2, we show that the value function is monotone in s_t^2. This means for any given action, in each decision epoch t, as the average capacity increases, the MDP value function will not decrease.

Theorem 2. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in* s_t^2.

In Theorem 3, we prove that the value function is monotone in s_t^1. This means for any given action in each decision epoch t, as the number of fully-charged batteries increases, the MDP value function will not decrease. If the demand for the MDP model is governed by a non-increasing discrete distribution, this result is implied from the result of Widrick et al. (2018). However, we strengthen the result as we *do not* require a non-increasing discrete distribution in Theorem 3.

Theorem 3. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in* s_t^1.

When considering both dimensions simultaneously, in Theorem 4, we prove that the value function is monotone in (s_t^1, s_t^2).

Theorem 4. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in* (s_t^1, s_t^2).

In a multi-dimensional setting, we need to define the concepts of partial ordering and partially non-decreasing function, which are given by Definitions 2 and 3 in Appendix 3.B. We also refer the reader to Appendix 3.B for full details of the proofs of Theorems 2, 3, and 4.

3.5 Solution Methodology

This section presents the solution methods used to solve the stochastic Scheduling, Allocation, and Inventory Replenishment Problem (SAIRP). First, we briefly describe the dynamic programming solution methods with the backward induction (BI) approach to provide exact solutions when the problem is not large-scale. Next, we present the approximate dynamic programming methods to overcome the curses of dimensionality and yield high-quality solutions for the stochastic SAIRPs.

3.5.1 Exact Solution Method: Dynamic Programming

Backward induction (BI) is an exact solution method to find optimal policies for the Markov Decision Process (MDP) problems (Puterman, 2005). Our goal is to find the optimal policy π^* that maximizes the expected total reward given by Equation (3.11). We attribute the optimal value function, $V_t^*(s_t)$, to the optimal policy. We calculate the optimal value based on cumulative values of taking the best actions onward from decision epoch t to N when in state s_t at time t (see Equation (3.12)). We use Bellman equations as presented in Equation (3.12) to find optimal policies and corresponding optimal value functions for $t = 1, \dots, N-1$ and $s_t \in S$.

$$V_t(s_t) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p_t(j \mid s_t, a_t) u_{t+1}(j) \right\}. \tag{3.12}$$

The BI algorithm starts from $t = N$ and sets $V_N(s_N) = r_N(S_N)$ according to Equation (3.9). Then, it finds the actions that maximize $V_t(s_t)$ for every state s_t moving backward in time ($t = N-1,\ldots,1$) using Equation (3.13). The optimal expected total reward over the time horizon is $V_1^*(s_1)$ where s_1 is the state of the system at the first decision epoch.

$$a^*_{s_t,t} = \text{arg max}_{a_t \in A_{s_t}} \left\{ r_t(s_t,a_t) + \sum_{j \in S} p_t(j|s_t,a_t) u_{t+1}(j) \right\}. \tag{3.13}$$

Now, we clarify terms used in the remainder of this chapter. A *sample path of demand* is the collection of a realized demand (uncertainty element) per time period generated from a given probability distribution. A *sample path of state* is comprised of the collection of consecutive visited states, one per time period. To calculate the visited states, we need the decision rule returned by a solution method for all states or visited states over time, the *sample path of demand*, and the present state. Using this information, we use the state transition functions (Equations (3.3) and (3.5)) to calculate the *sample path of state*. A *sample path of policy* is the set of consecutive decision rules of the visited states of the system. We use the term *instance* to refer to an example of stochastic SAIRP, specifically when we discuss the size of stochastic SAIRPs. The term *scenario* is used to refer to examples within our space-filling designed experiment that include different values for parameters. These values are generated such that to cover the designed experiment space (see Section 3.6.3).

3.5.2 Approximate Dynamic Programming Solution Methods

In this section, we outline our monotone approximate dynamic programming algorithm with regression-based initialization (MADP-RB). Approximate dynamic programming is a proven solution method that overcomes the curses of dimensionality (Powell, 2011). Using the foundation of the monotone approximate dynamic programming algorithm proposed by Jiang and Powell (2015), we make enhancements by exploiting our theoretical results to intelligently approximate the initial value function approximation and update the approximation with each algorithmic iteration.

The stochastic SAIRP suffers from the curses of dimensionality considering the size of all MDP elements together. In Chapter 2, we showed that the size of the the state space, the action space, the transition probability function, and the optimal policy are $O(\frac{M(1-\theta)}{\varepsilon})$, $O(M^2)$, $O(M^4 N(\frac{1-\theta}{\varepsilon})^2)$, and $O(MN\frac{1-\theta}{\varepsilon})$, respectively, where M, N, ε, and θ are the number of batteries, the time horizon, the capacity increment, and the replacement threshold, respectively. For instance, the size of the transition probability function is $O(10^{15})$ for a realistic-sized problem with $M = 100$ batteries, planning over a one month time horizon in one hour increments $N = 744$ with discretized battery capacity in increments of $\varepsilon = 0.001$. Due to these large sizes, standard MDP solution methods, such as backward induction (BI), were ineffective in solving realistic-sized instances of the stochastic SAIRP (see Chapter 2). Although there are many different ADP algorithms and approaches (Powell, 2011), there is no standard method to link the best algorithm to solve any particular problem. However, using the problem structure is good practice when developing efficient and effective algorithms. As we proved in Theorems 2, 3, and 4, the value function is monotonically non-decreasing in both dimensions, s_t^1 and s_t^2. Hence, it is reasonable to utilize and enhance the monotone approximate dynamic programming algorithm proposed by Jiang and Powell (2015). This algorithm has already shown promising performance for several application areas (Jiang and Powell, 2015). We proceed by outlining the core steps of (Jiang and Powell, 2015)'s monotone approximate dynamic programming (MADP) algorithm while highlighting our additions and changes to create the monotone approximate dynamic programming algorithm with regression-based initialization (MADP-RB). To aid with the explanation, in Algorithm 3, we outline the MADP and underline the enhancements for our MADP-RB. First, we introduce the notation necessary for the ADP algorithms in Table 3.1.

Table 3.1: Notation used in the ADP algorithms.

Notation	Description
$maxIteration+1$	The maximum number of regression-based initialization iterations
$\overline{M}$	The starting number of batteries used for the small SAIRPs solved using BI
$\overline{T}$	The time horizon in the small SAIRP
$u_t^{iter}(s_t)$	The optimal value of being in state s_t at iteration $iter$ and time t
τ	The maximum number of core ADP iterations
$V_t^n(s_t)$	The optimal value of being in state s_t at time t for iteration n
$\overline{V}_t^n(s_t)$	The approximate value of being in state s_t at time t for iteration n
$\hat{\upsilon}_t^n(s_t^n)$	The observed value of state s_t^n at time t for iteration n
$z_t^n(s_t^n)$	The smoothed value of being in state s_t at time t for iteration n

3.5.2.1 Monotone ADP with Regression-Based Initialization

In this section, we describe the core steps of the MADP algorithm proposed by Jiang and Powell (2015) and our enhancement using regression-based initialization in Algorithm 3. We display our enhancements in Algorithm 3 with underlines to make it more clear for the reader. We proceed by explaining the implementation of the algorithm.

The Monotone ADP with Regression-Based Initialization (MADP-RB) has two main steps. In the first step, we intelligently initialize the value function approximation using a linear regression function. The coefficients of the regression function are derived from feeding the optimal solutions of small SAIRPs. The second step is the core MADP algorithm that consists of updating the approximated values of visited and non-visited states over time through an iterative process. The states are visited over time at each iteration using the information of the present state, realized uncertainty, and taken action. The approximated value of the visited state is updated based on the observed value and previous approximated value. At each iteration, the monotonicity operator updates the approximated value of the non-visited states over time. We proceed by explaining each step in detail.

The first step of the MADP algorithm is to initialize the value function approximation for all decision epochs such that the monotonicity of the value function is preserved. Commonly, this is done by assigning a constant value, e.g., 0, to $\overline{V}_t^0(s_t)$ for all $s_t \in S$ and $t = 1, \ldots N-1$. However, using 0 or any constant value fails to exploit how the monotone value function changes based on state and time. Thus, our enhancement to the MADP algorithm is to intelligently approximate the initial value function approximation by exploiting the monotonicity of the value function. To do so, we iteratively calculate the optimal value function for small but increasing problem instances. Then, we use linear regression to approximate the initial value function for larger problem instances. To further explain this enhancement, we outline how these steps can be applied to the stochastic SAIRP.

In the stochastic SAIRP, as M and T increase, the problem suffers from the curses of dimensionality. For instance, the transition probability is $O(10^{15})$ for a realistic-sized problem when $M = 100$, $N = 744$, and $\varepsilon = 0.001$. Thus, we first optimally solve small instances of the stochastic SAIRP with small numbers of batteries $\overline{M} \ll M$ and time periods $\overline{T} \ll T$. We repeat this step by slowly increasing the number of batteries by one until we reach a user defined maximum number of iterations (or until it is computationally infeasible to optimally solve small instances with BI). For each instance, we determine the optimal value function and the associated trends based on changes in the state, time, and number of batteries. From the value functions for smaller instances of the problem, we use linear regression on the decision epoch (t), the state of the system (s_t), and the number of batteries in the station (M) to approximate the initial value function approximation for larger problem instances. See lines 1-6 in Algorithm 3 that describe this regression-based enhancement which are new and distinct from Jiang and Powell (2015). We complete the initialization phase in line 7, where we set the approximate value for the terminal epoch for all states and all core iterations $n = 1, \ldots, \tau$ to the terminal reward (see Equation (3.9)).

The algorithm iteratively proceeds in accordance with (Jiang and Powell, 2015) in lines 8-12. With each iteration, we select the best action starting from a random initial state (line 10) and move forward in time (line 11) using a sample observation of uncertainty and the approximate value of the future state, $\overline{V}_{t+1}^{n-1}(j)$ (line 12). Specifically, we pick the best action (line 12)

Algorithm 3 Monotone Approximate Dynamic Program with Regression-based Initilization

```
1: Initialize $\overline{M}$ batteries and $\overline{T}$ time periods, where $\overline{M} \ll M$ and $\overline{T} \ll T$.
2: Set iteration counter $iter = 0$.
3: for $iter \leq MaxIterations$ do
4:     $u_t^{iter}(s_t) \leftarrow$ Use backward induction to find the value function for the problem with $\overline{M} + iter$ batteries and $\overline{T}$ time periods, where $\overline{M} \ll M$ and $\overline{T} \ll T$.
5: end for
6: Initialize $\overline{V}_t^0(s_t)$ using linear regression on the combined $u_t^{iter}(s_t)$ for all states at $t = 1, \ldots, N-1$.
7: Set $\overline{V}_N^n(s) = r_N(s)$ for $s \in S$ and $n = 1, \ldots, \tau$
8: Set $n = 1$ $n \leq \tau$
9: Select initial state $S_1^n$
10: for $t = 1, \ldots, N-1$ do
11:     Sample an observation of the uncertainty, $D_t$, determine optimal action $a_t^n$ and future value $\hat{\upsilon}_t^n(s_t^n)$.
12:     Smooth the new observation with the previous value,
            $$z_t^n(s_t^n) = (1-\alpha_n)\overline{V}_t^{n-1}(s_t^n) + \alpha_n \hat{\upsilon}_t^n(s_t^n) \quad (3.14)$$
13:     Perform value function monotonicity projection operator as in Jiang and Powell (2015)
14:     Determine next state, $S_{t+1}^n$
15: end for
16: Increment $n = n + 1$
```

using Equation (3.15) and store the current observed value using Equation (3.16). In Equation (3.16), we use the previous approximation of the future states, $\overline{V}_{t+1}^{n-1}(s_{t+1})$, as the approximation of $\mathbb{E}(V_{t+1} \mid s_t, a_t)$.

$$a_{s_t,t}^n = \underset{a_t \in A_{s_t}}{\arg\max} \left\{ r_t(s_t, a_t) + \overline{V}_{t+1}^{n-1}(s_{t+1}) \right\}. \tag{3.15}$$

$$\hat{\upsilon}_t^n(s_t^n) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \mathbb{E}(V_{t+1} \mid s_t, a_t) \right\} \tag{3.16}$$

In line 13, we use a combination of the current value function approximation, $\overline{V}_t^{n-1}(s_t^n)$, and the current observed value, $\hat{\upsilon}_t^n(s_t^n)$ to calculate the smoothed value of being in state s_t, $z_t^n(s_t^n)$. This combination is weighted based on a stepsize function, α_n. Traditional stepsize functions, including $1/n$ (Powell, 2011) and harmonic (Powell, 2011; Rettke et al., 2016; Meissner and Senicheva, 2018), usually smooth the value function using pure observations for early iterations and gradually put less weight on the observation and put more on the approximations as the number of iterations increases.

Next, we apply the monotonicity projection operator as defined in Jiang and Powell (2015) (line 14). We note that the algorithm stores the value of all the states, and the monotonicity operator adjusts the value of non-visited states according to the value of the visited state. It ensures that no approximated value violates the monotonicity of the value function. For instance, consider the visited state s_t and an arbitrary state $\widetilde{s_t}$ such that $s_t \leq \widetilde{s_t}$ and $\overline{V}_t(s_t) > \overline{V}_t(\widetilde{s_t})$. The monotonicity operator increases the approximation for $\widetilde{s_t}$ up to $\overline{V}_t(s_t)$ and preserves the monotonicity property of the value function. Similarly, we decrease the approximation for lower states (i.e., $\widetilde{s_t} \leq s_t$) with higher value approximations than the visited state. Lastly, the algorithm moves forward in time to the next decision epoch until the last decision epoch, wherein a new iteration begins. Every new iteration starts from an arbitrary state and steps forward in time until the input number of iterations are completed.

3.5.2.2 Stepsize Function

Finding a good stepsize is a problem-dependent procedure that requires empirical experiments (Powell, 2011). A stepsize function, α_n is used to scale the current observed value and $(1-\alpha_n)$ is used to scale the current value function approximation. Because the value function approximation is often initially set to constant values and therefore is not informative, many stepsizes start with higher α_n values that emphasize the observed value. Then, as more iterations are conducted, α_n is decreased in order to place a greater emphasis on the value function approximation. We note that, a stepsize function needs to satisfy the three basic conditions given by Powell (2011) to guarantee convergence. A basic example of such stepsize function is $1/n$. Our preliminary experiments show that using the $1/n$ stepsize function is not appropriate as the rate of decreasing α_n over iterations is too fast for SAIRPs. Instead, we use the harmonic stepsize function (Powell, 2011) that uses a user-defined parameter, w, to controls the rate of decrease over iterations. In Equation (3.17), we provide the formal definition of the harmonic stepsize function.

$$\alpha_n = \frac{w}{w+n-1} \tag{3.17}$$

Additionally, we use the Search-Then-Converge (STC) stepsize rule that can control the rate of decrease in α_n by appropriately setting the parameter values (Powell, 2011). Hence, the STC function is suitable for cases like ours that need an extended learning phase (Powell, 2011). The STC rule was initially proposed by Darken and Moody (1992); however, we use the generalized STC formula given by George and Powell (2006) presented in Equation (3.18).

$$\alpha_n = \alpha_0 \frac{\left(\frac{\mu_2}{n} + \mu_1\right)}{\left(\frac{\mu_2}{n} + \mu_1 + n^{\zeta} - 1\right)} \tag{3.18}$$

The harmonic and STC stepsize functions follow the common process of weighting the observation higher earlier and decreasing this weighting with each iteration. The harmonic and STC stepsize functions are classified as deterministic as their values do not change based on the observations. In contrast, adaptive stepsize functions are sensitive to changes in the observations. To broaden our investigation, we also use the adaptive stepsize first introduced in George and Powell (2006) in Section 3.6.

3.6 Computational Results

In this section, we present the results and insights from computational experiments on different SAIRP instances. First, we present the data used to solve modest and realistic-sized SAIRP instances. We clarify how we distinguish modest, realistic-sized, and small SAIRP instances. We denote SAIRP instances as modest if they are optimally solvable using backward induction (BI) (i.e., 7 batteries total and only a one-week time horizon). We denote SAIRP instances as realistic-sized if they include larger numbers of batteries and the system is considered for longer time horizons (i.e., 100 batteries over a one-month time horizon). Due to the curses of dimensionality, the realistic-sized instances are not optimally solvable using BI. Within our monotone approximate dynamic programming with regression-based initialization solution procedure, we initially solve small SAIRP instances that are optimally solvable using BI in a matter of minutes (i.e., 2, 3, or 4 batteries over a one-week time horizon). After explaining the data for these instances, we proceed by explaining in detail the regression-based initialization used in MADP-RB, the Latin hypercube sampling (LHS) designed experiments, solution method comparison, and solutions/insights for both the modest and realistic-sized SAIRPs.

3.6.1 Explanation of Data

For the computational results, we use realistic data representing the costs to recharge, discharge, and replace a battery, the demand, and the battery degradation rate. To avoid redundancy in presenting similar insights for both EVs and drones, we focus on drones. First, we present the associated data, and then we show the results of a comprehensive set of experiments in Sections 3.6.3 and 3.6.4.

First, we set parameters associated with drone battery costs. We set the cost to recharge a battery using the historical power prices from the Capital Region, New York area in 2016 (National Grid, 2016). We use the time frame with the highest total power price for the modest and realistic-sized instances with the time horizon of a week and a month, respectively. Hence, as displayed in Figure 3.2, we select December 12-18 and the month of December for the modest and realistic-sized problems, using dashed and solid lines, respectively. We multiply these historical time-varying power prices by the maximum capacity of a battery to calculate the non-stationary, time-varying costs to recharge a battery. We assume a drone battery has a maximum capacity equal to 400 Wh as in the DJI Spreading Wing S1000 battery (DJI, 2019). We assume the revenue earned from discharging a battery back to the grid is equal to the charge price. Consistent with level 2 or 3 battery charging (Morrow et al., 2008; Ribbernick et al., 2015; Tesla, 2017), we assume a depleted (full) battery takes one hour (i.e., time between two consecutive decision epochs) of recharging (discharging) to become full (depleted). We use the purchase price for batteries to calculate the cost of battery replacement. Assuming the price per kWh of a battery is approximately \$235 (Romm, 2017), we set the replacement cost of a drone battery to be \$100. This calculated replacement cost serves as the baseline price, which is used and varied in the Latin hypercube designed experiments in Section 3.6.3.

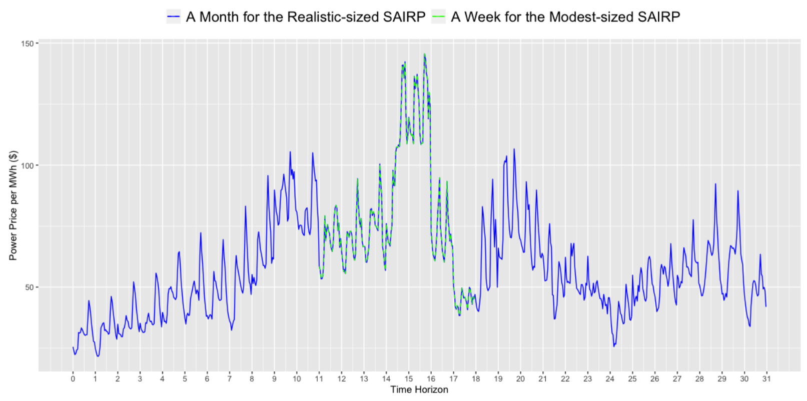


Figure 3.2: Power price fluctuations over December (realistic-sized SAIRP) and the week of Dec. 12-18 (modest Size SAIRP), 2016 in the Capital Region, New York.

In the absence of real data representing the number of customers demanding swaps at the station over time, we use the methodology of Chapter 2, Widrick et al. (2018), and Nurre et al. (2014) to derive the mean demand at the swap station over time. We assume the mean demand, λ_t, is equivalent to the historical arrival of customers at Chevron gas stations (Nexant, Inc. et al., 2008). Using λ_t, we assume the demand follows a Poisson distribution where t is the hour of the day. We scale λ_t to be in line with the number of batteries in the problem instance. Let λ_t' be the scaled demand for M' number of batteries. Because λ_t is originally used for $M = 7$, we calculate λ_t' values by multiplying $^{M'}/_{7}$ by λ_t. In Figure 3.3, we display the mean demand by hour over a one-week time horizon for the modest instances with $M = 7$. For longer time horizons, we assume the mean arrival of demand repeats every week.

We use existing studies to calculate the battery degradation rate per cycle. Although there are several factors that influence the battery degradation process, research states that the capacity fading has a linear behavior, especially in the first 500 cycles (Dubarry et al., 2011; Hussein, 2015; Lacey et al., 2013; Ribbernick et al., 2015). We note that the standard number of cycles

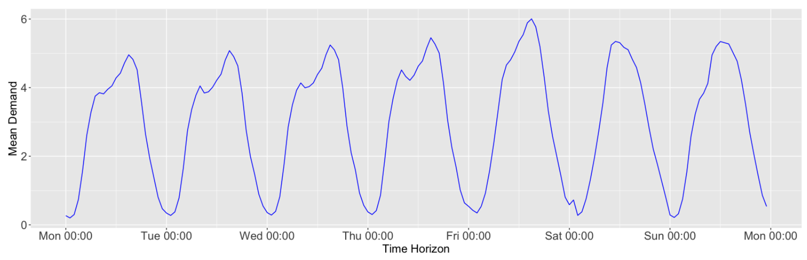


Figure 3.3: Mean demand for swaps over time.

for a drone Lithium-ion battery is 300 to 500 ($\delta^C \approx 0.1\%$) (Battery University, 2017). We select a higher value for the degradation rate to account for the elements that accelerate the degradation process, such as temperature from continuous use and recharging in swap stations. Thus, we select $\delta^C = 2\%$ as the baseline degradation rate and vary this value in our computational experiments to capture how changes in the degradation rate impact the policy and performance of the system. We note, the model is robust in that future experiments can be conducted for different baseline δ^C values to represent different degradation characteristics. In general, the industry-accepted battery end of life value is 80% of capacity. As follows, we set the replacement threshold $\theta = 80\%$ (Wood et al., 2011; Debnath et al., 2014). We note that $s_t^2 < 80\%$ is equivalent to the absorbing state of the system $s_t^2 = 0$ wherein the feasible action set is $A_{(s_t^1,0)} = \{(0,0)\}$. Hence, we can only replace batteries when the average capacity of batteries is not less than 80%.

3.6.2 Regression-Based Initialization

In this section, we explain our regression-based method to initialize the MADP-RB intelligently. We examine the empirical experiments of small SAIRPs and detect that the value function of the optimal policies $V_t(s_t)$ is a function of the decision epoch (t), the state of the system (s_t), and the number of batteries in the station (M). To clarify, we display the optimal values $V_t(s_t)$ of scenario 6 from the Latin hypercube designed experiments presented in Section 3.6.3. We display two consecutive decision epochs in Figure 3.4. In this example, we compute and show the optimal $V_t(s_t)$ when we solve small SAIRPs with $M = 2,3,4$.

As shown in Figure 3.4, the horizontal axis denotes the states, and the vertical axis shows the corresponding values. Among various techniques to estimate or forecast $V_t^0(s_t)$, we want a fast and simple method to generate and assign the initial approximations. Therefore, we propose using the linear regression function presented in Equation (3.19) to initialize the approximated value function, $\overline{V}_t^0(s_t)$.

$$\overline{V}_t^0(s_t) = h_0 + h_1 M + h_2 s_t^1 + h_3 s_t^2 + h_4 t \tag{3.19}$$

Having solved the small SAIRPs, we find the appropriate values for h_0, h_1, h_2, h_3, and h_4. In Sections 3.6.3 and 3.6.4, we demonstrate how the intelligent initial value function approximation leads to superior results.

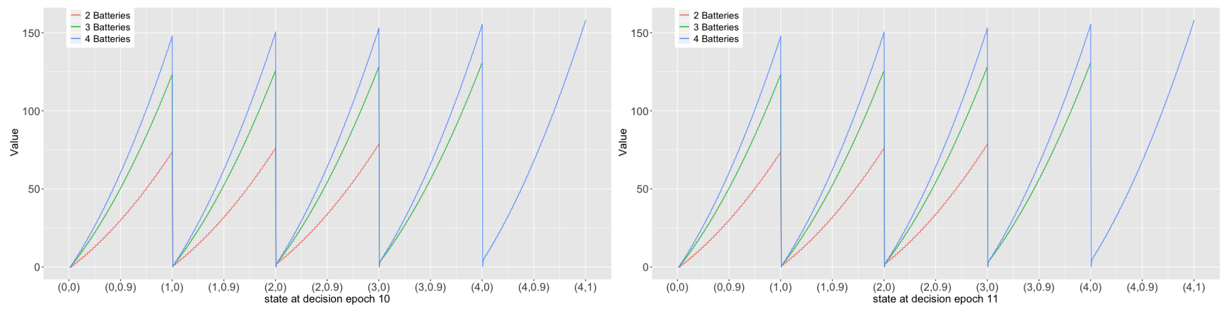


Figure 3.4: An instance of the optimal values over states in two consecutive decision epochs

3.6.3 Latin Hypercube Designed Experiments for Modest SAIRPs

In this section, we perform a Latin hypercube sampling designed experiment that is used to assess the quality of the solution methods and to deduce insights for the battery swap station application. A Latin hypercube sampling (LHS) designed experiment is a space-filling design with broad application in computer simulation (Montgomery, 2008). Using the LHS configuration of Chapter 2, with the test set of 40 scenarios generated to cover the design space of parameters, we first quantify the performance of MBI, MADP, and MADP-RB.

In Chapter 2, we were able to optimally solve 40 modest LHS scenarios using BI. We run all computational tests using a high-performance computer with four shared memory quad Xeon octa-core 2.4 GHz E5-4640 processors and 768GB of memory. Even on a high-performance computer, the largest scenario we were able to optimally solve using BI is with $M = 7$, $\varepsilon = 0.001$, and $N = 168$ which represents a full week of operations where each decision epoch is one hour. Using these modest scenarios, herein, we are specifically concerned with quantifying the speed and performance of MBI, MADP, and MADP-RB against a known optimal policy and optimal expected total reward. The factors and their associated lower and upper bounds are defined as in Table 3.3 (we refer the reader to Chapter 2 for a complete justification for how these low and high values are calculated).

Memory Intensive Processing vs. Compute Intensive Processing. We can implement BI and MBI in two ways. In the first way, which we denote *'Memory Intensive'*, we calculate and store all of the transition probability values, so they are readily available throughout the execution of the algorithm. In the second way, which we denote *'Compute Intensive'*, we do not store any probability values and instead calculate each probability value when needed for calculations through the execution of the algorithm. As is indicated in their names, the *Memory Intensive* way requires more memory and the *Compute Intensive* way requires more time as it needs to calculate probability values numerous times. We implement and use both ways to solve the modest SAIRP instances on a high-performance computer (HPC). We note, on this HPC, we have different options. If we want to run the algorithm for longer, we have up to 72 hours of computational time available per run with access to only 768 GB of memory using four shared memory quad Xeon octa-core 2.4 GHz E5-4640 processors. However, if we want to use more memory, we have access to 3 TB of memory (four shared memory Intel(R) Xeon(R) CPU E7-4860 v2 2.60GHz processors) but only 6 hours of computational time. Thus, our results reflect these limitations.

In Table 3.2, we report the memory used and computation times for BI and MBI by instance and method. We use red to highlight the instances where we either exceeded the memory or time available and thus, do not find a solution. As is evident by the results, BI and MBI are not tractable solution methods for even modest-sized SAIRPs. MBI does outperform BI; however, the computation time is significant even for $M = 10$. Although not shown in Table 3.2, when $M \geq 12$, MBI exceeds the 72-hour time limit. We note that our approximate solution methods do not run into memory issues. Hence, we move forward with the faster processing method, *Memory*

Intensive, to solve SAIRPs hereafter.

Table 3.2: Time and memory used for different size of SAIRPs using memory intensive and compute intensive methods.

	Memory Intensive			Compute Intensive		
Size	Mem. Used (GB)	BI Comput. Time (h)	MBI Comput. Time (h)	Mem. Used (GB)	BI Comput. Time (h)	MBI Comput. Time (h)
$M = 7$	830	3.8	3.6	0.2	29.2	7.5
$M = 8$	1120	> 6	> 6	0.2	52.6	11.3
$M = 9$	2030	> 6	> 6	0.2	> 72	16.3
$M = 10$	> 3072	> 6	> 6	0.2	> 72	24.8

We computationally test MBI for the general case when no limits are placed on how many batteries are replaced per decision epoch and the MADP and MADP-RB methods with the harmonic, Search-Then-Converge (STC), and adaptive stepsize functions. For the core ADP procedure, we run the scenarios for $\tau = 500000$ iterations. For these stepsize functions, based on extensive computational experiments, we present the most favorable results. For the harmonic stepsize function, Powell (2011) recommends that $\alpha_n < 0.05$ in the last iteration ($\tau = 500000$). Thus, we set $\alpha_\tau = 0.05$ and $w = 25000$. We note, our tests show that the average optimality gap increases when w is below 25000, thus we use $w = 25000$. For setting the STC parameters, we let $\alpha_0 = 1$ to put the total weight on the observation at the beginning of the procedure. This results in $\alpha_1 = 1$. We test six values for μ_1 = 10, 100, 500, 600, 1000, and 10000, five values for μ_2 = 10, 100, 1000, 10000, and 100000, and 3 values for ζ = 0.6, 0.7, and 0.8 to adjust the tuple of parameters (μ_1, μ_2, ζ). Over these 90 experiments, we observe that (μ_1, μ_2, ζ) = (600, 1000, 0.7) yields the best result in terms of the average optimality gap. Thus, we use this tuple of parameter values for the STC stepsize. For the adaptive stepsize function, we use the setting presented in Powell (2011). In the adaptive stepsize, we need to estimate the bias and its variance. For smoothing the observation and approximation of the bias and its variance, Powell (2011) recommends using a harmonic stepsize that tends to a value between 0.05 and 0.1 in the last iteration. Consistent with this recommendation, we use the harmonic stepsize function with $w = 25000$. For the MADP-RB, we set $\overline{M} = 2$, $\overline{T} = T = 168$ hours, MaxIterations = 2 (i.e., 3 iterations are used as the input for the linear regression and value of *iter* can be 0, 1, and 2), and use BI to optimally solve the scenarios for 2, 3, and 4 batteries. We then plug these results into Equation (3.19) to set the initial approximation as in line 6 of Algorithm 3.

Table 3.3: Factors with associated low and high values for use in the Latin hypercube designed experiment.

Factor	Low	High
Basic revenue per swap (β)	1	3
Replacement cost L_t	2	100
Battery degradation factor (δ^C)	0.005	0.02

Later, we will show that adding the regression-based initialization to MADP significantly improves the quality of solutions. One can argue that initialization using any monotone value function leads to the same result. We tested this argument using a set of arbitrary monotone value functions and report the best-founded function. We call the algorithm with this arbitrary monotone value function initialization MADP-M. We note that in MADP, the initial value function is a constant and equals to $\overline{V}_t^0(s_t) = 0 \; \forall s_t \in S$. In MADP-M, the monotone value function used for initialization is $\overline{V}_t^0(s_t) = \left(\frac{\beta(1+s_t^2-2\theta)}{1-\theta}\right)(s_t^1) + k(N-t)$ where k is a constant. The first term is a monotone value function that equals the revenue generated from swapping all of the full batteries when in state s_t at time t. The second term is a non-negative term with an opposite relationship with time, which means the value of being in every state, s_t, is higher at earlier decision epochs. The logic behind adding the non-negative term is as follows. The value of being in state s_t, $V_t(s_t)$, accumulates the reward from time t onward to the end of the time horizon, so we may expect to gain a higher reward (profit) over a longer time. Although $V_t(s_t)$ is not always decreasing in t, the general trend is observed in the optimal values of small SAIRPs. Our tests show that MADP-M with $k \leq 1$ outperforms MADP. Then, we test $k = 0.1, 0.2, \ldots, 1$ and observe that MADP-M with $k = 0.5$ yields the best result.

In Table 3.4, we summarize the 40 scenarios and the results comparing the approximate methods to backward induction. Specifically, we calculate an *average optimality gap over all* *scenarios*, comparing MADP (monotone approximate dynamic programming), MADP-M (monotone approximate dynamic programming algorithm with arbitrary monotone value function initialization), MADP-RB (monotone approximate dynamic programming algorithm with regression-based initialization), and Monotone Backward Induction (MBI) to the optimal expected total reward calculated using BI. In Equation (3.20) for each scenario of MADP, MADP-M, and MADP-RB, we input the expected total reward of the last iteration into *the expected reward of the approximated method*. Then, we can calculate the average and maximum optimality gaps over 40 scenarios given by the last two rows of Table 3.4.

$$\text{Optimality Gap} = \left| \frac{\text{Expected Reward BI - Expected Reward Approximated Method}}{\text{Expected Reward BI}} \right| * 100\% \tag{3.20}$$

Immediately evident from Table 3.4 is that MADP-RB led to significantly smaller average and maximum optimality gaps than Jiang and Powell (2015)'s MADP, MADP-M, and MBI. Overall, we find that MADP-M outperforms MADP in regards to the average optimality gap. This demonstrates that initializing an ADP approach, even with an arbitrarily monotone value function does result in benefit. However, we further find that when considering both the average and maximum optimality gaps, MADP-RB significantly outperforms MADP-M. This demonstrates the significant benefit of the regression-based initialization. We proceed by further analysis of MADP and MADP-RB.

We note, MBI is a reasonable approximation for scenarios with high replacement costs, but it does not provide competitive optimality gaps when replacement cost is low. However, when we limit the number of batteries replaced in each epoch in accordance with Theorem 1, the optimality gap is 0.00%. MBI is smarter than BI and can save computational time when searching for the best policies. In other words, we do not need to loop over all the actions when using the monotonicity property of the optimal policy. However, as we showed, BI and even MBI are not computationally tractable for realistically sized instances of SAIRPs, which necessitates using approximate solution methods.

Table 3.4: Optimality gap (%) of MADP, MADP-M, MADP-RB, and MBI.

				Harmonic Stepsize			STC Stepsize			Adaptive Stepsize			MBI
Scen.	β	L_t	δ^c	MADP	MADP-M	MADP-RB	MADP	MADP-M	MADP-RB	MADP	MADP-M	MADP-RB	
1	1.03	45	0.009	27.11	6.66	8.26	30.67	8.57	10.45	20.00	7.71	8.77	0.00
2	1.08	82	0.011	14.14	13.68	12.10	16.09	12.19	10.58	7.67	3.90	2.40	0.09
3	1.13	61	0.005	43.50	12.14	6.27	47.01	17.28	7.54	11.69	8.31	7.38	6.05
4	1.20	69	0.009	26.08	8.28	1.26	30.10	10.24	3.00	19.84	8.81	3.34	0.00
5	1.23	89	0.008	14.80	2.20	8.62	18.62	0.53	7.88	4.61	0.15	1.73	0.00
6	1.26	47	0.010	27.76	12.06	9.72	30.10	13.51	10.95	19.66	12.37	9.59	0.00
7	1.32	98	0.019	12.11	22.34	9.15	13.73	22.8	9.32	9.07	4.17	0.71	0.00
8	1.39	94	0.011	13.49	8.38	12.76	16.33	7.97	10.82	7.28	2.56	2.68	0.00
9	1.41	87	0.014	6.93	19.06	15.37	9.39	18.87	14.55	4.33	6.35	3.31	0.00
10	1.48	36	0.006	25.41	15.61	9.20	33.78	15.97	9.58	26.50	14.75	3.47	7.25
11	1.53	68	0.016	17.28	7.24	3.17	18.05	6.56	1.60	13.51	1.43	0.94	0.00
12	1.56	28	0.018	15.04	2.22	1.25	15.37	0.78	0.06	12.79	6.46	6.11	5.95
13	1.60	57	0.010	27.51	14.45	10.10	31.56	15.74	11.84	20.77	14.18	1.46	0.00
14	1.68	31	0.017	30.02	15.21	6.18	28.94	15.29	4.92	23.95	14.52	10.45	11.05
15	1.71	62	0.006	26.79	10.13	5.25	33.69	10.78	4.62	15.92	8.63	1.19	0.00
16	1.76	44	0.017	19.14	0.36	1.14	20.91	0.32	0.20	14.72	3.62	2.88	0.00
17	1.83	55	0.016	17.36	3.00	1.84	17.63	2.39	1.13	13.29	1.46	1.83	0.00
18	1.88	51	0.019	11.38	10.00	7.55	13.74	10.51	6.57	8.32	1.32	1.00	0.00
19	1.94	66	0.012	17.12	2.01	3.99	18.23	2.35	3.06	11.17	2.91	0.17	0.00
20	1.95	73	0.019	11.22	11.59	9.15	13.51	10.74	8.47	9.27	1.76	0.82	0.00
21	2.00	83	0.012	16.41	1.95	4.27	19.50	3.06	3.95	11.45	3.29	0.01	0.00
22	2.07	20	0.013	60.56	52.87	0.38	53.56	54.87	1.59	56.97	54.10	0.39	53.97
23	2.15	90	0.013	13.40	3.44	12.19	15.41	2.44	11.46	9.88	0.27	2.06	0.00
24	2.16	41	0.007	33.37	31.29	2.02	27.58	29.17	3.07	35.00	33.40	0.94	30.69
25	2.22	79	0.015	7.42	7.37	12.45	10.11	7.45	12.03	5.07	3.08	1.55	0.00
26	2.27	8	0.007	64.60	67.51	8.99	68.96	70.09	8.91	66.91	70.85	8.79	47.85
27	2.32	25	0.017	59.25	45.07	4.37	59.72	40.87	5.03	44.85	44.52	5.13	49.80
28	2.37	74	0.014	8.40	8.27	13.87	8.27	7.62	12.80	4.45	3.34	2.95	0.00
29	2.40	7	0.018	68.69	69.38	15.65	72.46	72.84	15.22	75.11	73.61	15.47	74.43
30	2.50	14	0.008	65.68	68.12	7.05	68.96	67.59	2.48	64.64	68.68	7.38	43.56
31	2.51	24	0.015	59.93	54.56	2.08	48.80	54.09	1.12	59.03	51.93	3.00	56.76
32	2.56	34	0.009	49.50	41.41	1.64	36.87	42.90	2.67	49.92	38.64	0.25	30.92
33	2.64	39	0.010	52.19	34.27	7.55	49.33	36.56	6.50	37.41	32.55	7.73	25.82
34	2.68	54	0.016	23.60	11.83	8.06	23.82	12.74	8.13	19.71	12.34	9.42	7.20
35	2.71	3	0.007	31.16	34.05	7.69	31.22	33.44	7.45	33.33	33.40	7.69	53.11
36	2.77	16	0.012	68.79	69.50	6.64	71.81	68.35	6.62	68.49	66.33	6.29	56.91
37	2.83	21	0.013	69.68	64.86	2.29	69.44	63.10	3.12	67.79	63.29	3.33	58.57
38	2.90	77	0.015	8.18	4.20	5.16	9.62	3.62	5.48	4.58	1.10	1.14	0.00
39	2.91	96	0.020	10.27	7.51	11.55	10.91	7.61	11.18	7.05	1.17	1.51	0.00
40	2.96	12	0.005	59.05	66.65	7.42	63.61	67.87	6.93	65.77	68.09	7.69	17.20
Avg Gap				30.86	23.52	7.09	31.94	23.74	6.82	26.54	21.23	4.07	15.93
Max Gap				69.68	69.50	15.65	72.46	72.84	15.22	75.11	73.61	15.47	74.43

As is expected, ADP methods take significantly less time than BI and MBI, which take on average 3.8 hours and 3.6 hours, respectively. The computational time of MADP-RB includes three major operations in Algorithm 3: (i) obtaining the data for the regression function in steps 1-5; (ii) deriving the regression function in step 6; and (iii) calculating the monotone ADP results in steps 7-18. The average computational time for operations (i), (ii), and (iii) are 902 seconds (0.25 hours), less than 1 second, and 1025 seconds (0.28 hours), respectively. We note that (iii) is equivalent to the average computational time of MADP without the regression-based initialization. In other words, if we execute the initialization steps offline, there is no difference between the computational time of MADP and MADP-RB as finding the coefficients of the regression function takes less than a second. As the problem instance of the SAIRP increases significantly, the required memory and time for BI and MBI also increase significantly. However, MADP and MADP-RB enable us to solve realistic-sized SAIRP instances, as we demonstrate in Section 3.6.4.

Next, to compare the two ADP methods, we quantify a measure of convergence. One way to evaluate an ADP method is to focus not only on the ending performance but also on the convergence with each iteration. Thus, we calculate *the average optimality gap over all iterations*, g_m^s, for each method m and scenario s. In this calculation, we average the optimality gap for all iterations (τ) as in Equation (3.21).

$$g_m^s = \frac{1}{\tau}\sum_{i=1}^{\tau} \frac{\left|\text{Expected total reward at iteration } i - \text{Optimal expected total reward}\right|}{\text{Optimal expected total reward}} * 100\% \quad (3.21)$$

In Figure 3.5, we show the expected total reward (value) as a function of the number of iterations for MADP and MADP-RB with the three different stepsizes. The rows of Figure 3.5 correspond to the harmonic, STC, and adaptive stepsizes, respectively. The columns correspond to the scenarios which result in the low, average, and high g_m^s in the LHS where m includes MADP with the three stepsizes. Within each subfigure, we display three lines showing the progression

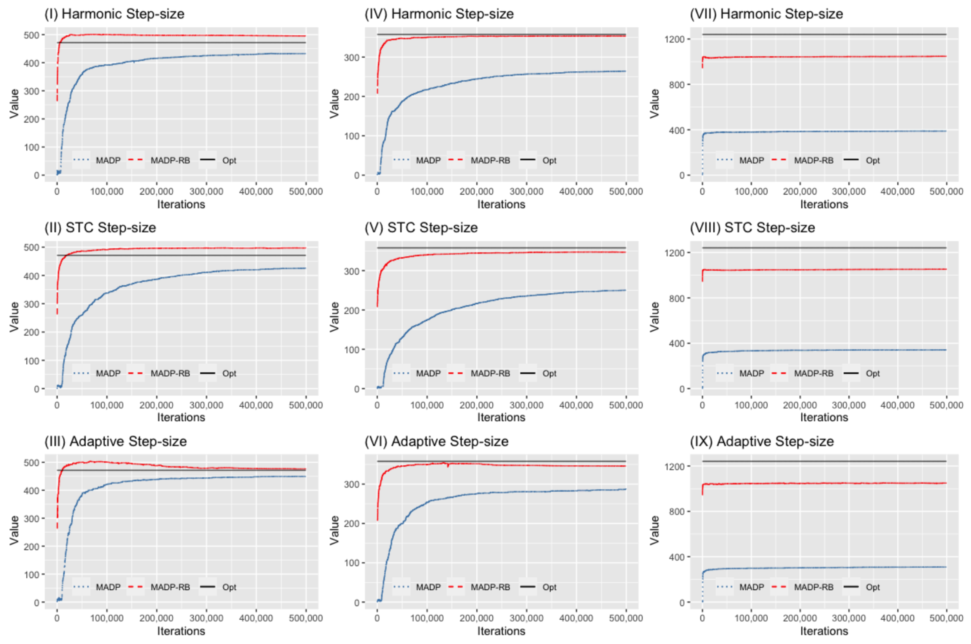


Figure 3.5: Expected total reward convergence of MADP and MADP-RB using different Stepsize functions vs optimal expected reward.

of the value by iteration for MADP and MADP-RB and the optimal value for the state s_1, which corresponds to the initial state of the swap station when all batteries are full and new with full capacity. From the subfigures, we observe that MADP-RB consistently outperforms MADP in regards to convergence.

Now, we explain the possibility of convergence to a value greater than optimal (see Figures 3.5-(I), 3.5-(II), and 3.5-(III)). The approximated value depends on two factors: (i) the previous approximation value and (ii) the present observation value. These values are smoothed using a stepsize function and set equal to the approximation value for the next iteration. Hence, as either value can be greater than the optimal value in any iteration, the smoothed value can take a value higher than the optimal value. For example, suppose the initial approximation is too high. In that case, the smoothed value can be higher than the expected value of being in any state.

We also calculate the *average optimality gap over iterations over scenarios* for each method m using Equation (3.22). In this equation, m = MADP and MADP-RB, and the total number of scenarios in our Latin Hypercube designed experiments is 40.

$$\overline{g_m} = \frac{\sum_{s=1}^{40} g_m^s}{40} \tag{3.22}$$

In Table 3.5, we display the *average optimality gap over iterations over scenarios* for MADP and MADP-RB using different stepsize functions given by Equation (3.22). As shown in Table 3.5, the average optimality gap associated with MADP-RB for all stepsizes is remarkably lower than MADP. This indicates that MADP-RB is able to more quickly converge to the optimal value as compared to MADP.

Table 3.5: Comparison between convergence over iterations using MADP and MADP-RB.

Average Optimality Gap over Iterations over Scenarios (%)					
MADP			MADP-RB		
Harmonic	STC	Adaptive	Harmonic	STC	Adaptive
36.72	40.84	32.58	6.89	8.53	5.90

The required computation effort for both ADP approaches depends on, in part, the number of iterations, τ. Thus, we analyze how the value function changes by iteration, which will inform our choice of τ for future experiments. In Figure 3.5, we observe the value function plateaus before $\tau = 500000$. Furthermore, we notice that both methods do not significantly change after $\tau = 100000$ in many scenarios. When examining all scenarios solved by MARP-RB, we calculate the percentage difference between the value at $\tau = 100000$ and $\tau = 500000$. For the harmonic stepsize, the average percentage difference is 1.03% and the maximum percentage difference is 8.86%. We obtained similar values for the STC and adaptive stepsize and scenarios; however, to avoid redundancy, we proceed with an analysis of the harmonic stepsize.

As the value does not always convey the operational changes in the policy which decision makers implement, we proceed by analyzing the changes in policy at iteration 100000 and 500000. We use Equation (3.23) to compute the percentage difference in the policies using at iteration 500000 and 100000 when 500 sample paths of realized demand generated. With this equation, we calculate the percentage difference between the summation of the recharging, discharging, and replacement actions as we decrease the number of iterations for each scenario. We then average this value over all scenarios. Specifically, in Equation (3.23), TC_s^a and $\overline{TC_s^a}$ represent the total number of actions, recharging/discharging ($a = a^1$) and replacements ($a = a^2$), for scenario s using $\tau = 500000$ and $\tau = 100000$, respectively. That is, $TC_s^{a^1}$ and $\overline{TC_s^{a^1}}$ denote the total number of recharging/discharging actions and $TC_s^{a^2}$ and $\overline{TC_s^{a^2}}$ are the total number of replacements for scenario s using 500000 and 100000 iterations, respectively. Our results show the average percentage difference in the number of recharged/discharged and replaced batteries over 40 scenarios are 6% and 5%, respectively.

$$\text{Average (\%) difference for action } a \text{ over all scenarios} = \frac{1}{40}\sum_{s=1}^{40}\frac{\left|\overline{TC_s^a} - TC_s^a\right|}{TC_s^a} * 100\%. \quad (3.23)$$

Thus, as the changes in the value and policies are not significant, we use $\tau = 100000$ in our future computational experiments for solving realistic-sized SAIRPs in Section 3.6.4.

Exploiting the monotonicity property does significantly improve the quality of solutions and convergence of ADP methods. To demonstrate this result, we run two versions of standard approximate value iteration (AVI) without and with regression-based initialization, AVI and AVI-RB, respectively. Adding the monotonicity property to AVI and AVI-RB converts them to MADP and MADP-RB, respectively. In Table 3.6, we present a summary comparing AVI, MADP, AVI-RB, and MADP-RB using the harmonic stepsize function and 500000 iterations. In this table, we present the average optimality gap of AVI, MADP, AVI-RB, and MADP-RB over all 40 scenarios of our Latin hypercube designed experiments (see Table 3.4 for full details of these 40 scenarios). As is evident by these results, using the monotonicity property decreases the average optimality gap. However, we observe that the regression-based initialization is even more impactful than the monotonicity operator. By adding the monotonicity operator, the average optimality gap decreases by 10.31% (AVI to MADP) but by adding the regression-based initialization, the average optimality gap decreases by 26.42% (AVI to AVI-RB). The lowest average optimality gap occurs with MADP-RB that has both the monotonicity property and regression-based initialization.

Table 3.6: Comparison between the average optimality gap of different approximate solution methods.

Approximate Method	AVI	MADP	AVI-RB	MADP-RB
Average Optimality Gap (%)	41.17%	30.86%	14.75%	7.09%

3.6.4 Monotone ADP Results and Performance for Realistic-sized SAIRPs

In this next set of computational experiments, we focus on the ability to solve and deduce insights from realistic-sized stochastic SAIRPs. We proceed by determining the parameters, summarizing the results, including the expected total reward and computational times, presenting sample paths of policies, and analyzing the relationship between the outputs and inputs of the experiments.

For the test instances, we solve all 40 scenarios of the designed experiment presented in Section 3.6.3 using the realistic data summarized in Section 3.6.1. Specifically, we consider 100 batteries, a one-month time horizon with each decision epoch representing one hour, and scaled mean demand $\lambda_t' = (\lambda_t)(M'/7)$ where $M' = 100$ and λ_t is the original mean arrival of demand at time t in line with the original modest-sized problem with $M = 7$. Due to the curses of dimensionality, BI and MBI are not capable of solving these large problems; thus, we focus on the performance of MADP and MADP-RB with $\tau = 100000$ iterations.

In Table 3.7, we present the expected total reward and required computational time for the MADP and MADP-RB methods with harmonic, Search-Then-Converge (STC), and adaptive stepsizes. The average computational time of MADP-RB is only 8 minutes longer than MADP for all stepsize functions as a result of executing the regression-based initialization (see steps 1-6 in Algorithm 3). We highlight the highest expected reward for each row of Table 3.7. We acknowledge that the highest expected total reward does not always indicate the lowest optimality gap; however, we use this as a metric of comparison in the absence of being able to determine the optimal solution and value for these realistic-sized SAIRPs.

The results are very clear and consistent. The MADP-RB value is always greater than MADP for all stepsize functions and scenarios. Within MARP-RB, we observe that harmonic, STC, and adaptive stepsize generate the highest expected total reward in 11, 3, and 26 scenarios, respectively.

Table 3.7: Results of realistic-sized SAIRPs for the latin hypercube designed experiment.

	MADP Expected Total Reward ($)			MADP-RB Expected Total Reward ($)		
Scenario	Harmonic	STC	Adaptive	Harmonic	STC	Adaptive
1	3468.3	2791.7	3639.3	6964.3	7303.6	**7884.1**
2	5404.1	4900.6	4773.9	7073.1	**7622.2**	7549.9
3	9324.8	4219.4	8788.3	15443.0	15840.8	**16491.2**
4	5397.1	3890.3	5836.5	8684.0	9645.6	**9710.6**
5	4783.1	4923.2	4986.2	10308.5	10597.4	**11079.1**
6	4824.0	2363.0	5443.4	7545.4	8123.5	**8797.8**
7	3779.6	4415.5	4422.3	**5269.7**	5176.2	5212.2
8	4479.5	3401.0	3521.3	8128.5	9336.6	**9486.3**
9	2087.0	2546.6	2730.5	6848.6	7446.5	**7780.6**
10	8461.4	8736.6	7297.4	**22060.2**	19707.0	20229.4
11	3684.1	3385.4	3749.2	6581.5	7463.9	**7889.3**
12	2888.5	2842.8	3357.9	5903.6	**6052.8**	5900.4
13	6248.1	4061.9	5994.1	11620.6	12228.4	**12950.7**
14	3253.2	2578.2	3426.1	7176.3	7258.9	**7573.4**
15	9983.3	9573.4	8814.8	16484.4	16326.9	**16614.3**
16	4446.9	3631.8	4474.7	7177.7	**7417.1**	7268.3
17	4789.0	3875.1	5372.4	7804.2	8601.2	**8787.8**
18	3958.3	3634.7	4291.3	6900.5	6998.8	**7260.1**
19	5855.0	4561.5	6722.5	10514.8	10892.3	**12124.5**
20	4090.8	3676.9	2051.8	7415.2	7501.5	**7529.3**
21	7545.6	5562.7	7116.9	10685.1	12764.6	**13478.4**
22	5454.8	5539.8	5699.1	**33221.8**	31557.5	32379.8
23	5629.3	5707.7	7111.5	11848.4	12005.5	**12811.4**
24	10593.2	10698.7	11037.8	**34805.9**	32215.2	32399.8
25	4963.0	5349.2	7186.8	10505.9	10074.7	**11664.4**
26	51874.6	48577.1	50855.3	84745.6	83125.1	**85354.7**
27	5224.7	5512.7	4699.4	16876.2	16771.0	**17203.0**
28	6366.9	5006.7	6514.1	11356.8	12461.2	**12748.8**
29	39244.6	34101.7	37832.0	**71877.0**	70920.1	71396.5
30	25188.0	24640.2	26046.0	79376.8	77325.9	**80233.1**
31	5373.0	5971.0	6832.8	**33432.0**	31564.1	32693.5
32	9216.7	10299.8	10055.9	**44189.8**	42772.4	43355.7
33	9929.9	9783.1	10293.0	**29615.5**	28922.0	28786.7
34	5920.8	4409.0	6452.9	11341.8	12115.3	**13178.0**
35	100038.0	98060.7	98254.7	122397.0	122683.0	**124779.0**
36	18743.7	17610.1	16362.2	**73562.5**	72612.0	73060.5
37	8067.0	9426.7	12138.2	**60294.6**	58483.0	59488.7
38	7726.4	5789.2	8285.0	12137.7	14070.3	**14674.7**
39	5699.1	5683.8	5168.2	11029.9	10982.9	**11158.6**
40	68536.0	63309.2	62951.7	**112542.0**	111153.0	112073.0
Avg CPU Time (hours)	8.5	8.5	8.6	8.6	8.6	8.7

In addition to examining the expected total reward of the two ADP methods, we examine sample paths of the policies when 500 sample paths of realized demand are generated. Our results show that the average percentage of demand met levels off around when the number of sample paths is greater than 200. As a result and to be conservative, for all scenarios, we calculate the average of demand met for 500 sample paths. We observe that the average of demand met for 500 sample paths over all the scenarios under MADP-RB is 25%, 29%, and 19% more than MADP with harmonic, STC, and adaptive stepsize functions, respectively. The optimal policies are similar across these three stepsize functions, thus, we proceed by presenting further analysis for MADP-RB policy using the harmonic stepsize function.

From the sample paths, we observe three typical behaviors within all LHS scenarios. In Figure 3.6, we displayed examples of the fluctuations of the average capacity for these three categories when the realized demand equals mean demand. In Type 1, the average capacity remains over 95% percent throughout the time horizon (see Figure 3.6a). This is achieved with many replacement actions. The scenarios that exhibit Type 1 behavior have low replacement costs and high revenue per swap (including scenarios 26, 29, 30, 35, 36, 40, 37, and 22). In Type 2 (see Figure 3.6b), the average capacity does not stay as high as Type 1, but is maintained above 85% for the entire time horizon except the very end of the month. For 500 sample paths of demand, on average, in scenarios with Type 1 behavior, we observe replacement in 60% of the epochs and when a replacement occurs, 15% of the batteries are replaced. In contrast, in scenarios with Type 2 behavior, replacement occurs in 41% of the decision epochs and 7% of the batteries are replaced when a replacement occurs. The scenarios that exhibit Type 2 behavior have average replacement cost and revenue per swap values (includes scenarios 31, 27, 32, 24, 33, 10, and 15). In these scenarios, it is beneficial to recharge batteries to ensure demand is met, but the station allows battery capacity to degrade and does not replace batteries as frequently. Finally, in Type 3 (see Figure 3.6c), the average battery capacity consistently decreases over the time horizon to the replacement threshold of 80%. Fewer batteries are replaced as, on average, we observe replacement in 11% of the decision epochs and only 3% of batteries are replaced when a replacement occurs. This demonstrates a reduced priority for maintaining a high battery capacity. At epochs when batteries are replaced, a small number of batteries are replaced at a time and are done to ensure the minimum capacity is maintained. The scenarios that exhibit Type 3 behavior have high replacement costs and low swapping revenue values.

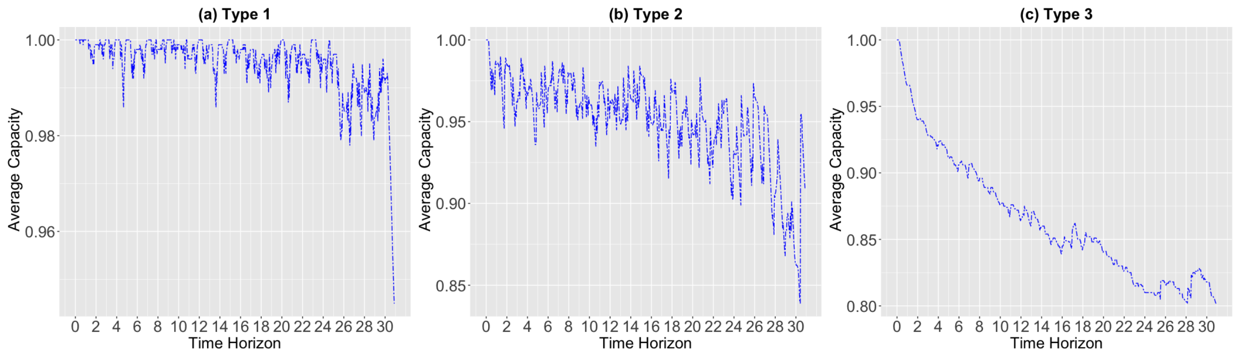


Figure 3.6: Sample paths of average capacity over time horizon for three types of LHS scenario when realized demand equals means demand.

Next, we dig into a specific scenario to compare and contrast the policies for the modest and realistic-sized instances. Our intent in this analysis is to determine whether similar actions are taken for the same scenario when considering more batteries over a long time horizon. We observed this is not the case, thereby justifying the need to solve realistic-sized instances. To demonstrate our observations, we present results for scenario 15 that has meaningful differences with regard to the replacement actions, charging actions, and amount of demand met between the modest and realistic-sized instances. In the realistic-sized problem, on average, we observe replacement in 27% of the decision epochs, and when replacement occurs 4% of the batteries are replaced. However, the derived policy in the modest problem consists of no replacement actions. With regard to charging, on average, we see charging actions in 97% and 40% of decision epochs for the realistic-sized and modest problems, respectively. With more frequent charging and replacement actions in the realistic-sized problem, the percentage of met demand is higher than the modest problem. The derived policy for the realistic-sized problem satisfies 76% of demand, which is significantly higher than 54% of the demand met for the modest problem. From this analysis, while we must solve a finite horizon MDP to capture the time-varying elements, it is necessary to be able to computationally solve instances with longer time horizons and a larger number of batteries that mimic reality in order to determine the general operating policies.

In Figure 3.7, we present a sample path of the policy when realized demand equals the mean demand for the realistic-sized SAIRP associated with scenario 15. We observe that more batteries are replaced before epochs with high power prices (e.g., 53% on day 12 before the high power price days of December 12-18), which is consistent among Type 2 scenarios. When this replacement occurs, the battery capacity is higher, which results in higher swapping revenue to offset to higher costs to recharge batteries. Overall, we observe a consistent trend for recharging batteries and meeting demand during the time horizon. The number of full batteries is kept between 40% and 60% of the total number of batteries during the middle of the day and recharging is conducted to raise the number of full batteries to 90% over night. We show in Figure 3.8 that the policy meets 100% of demand during off-peak epochs and more than 50% during peak epochs.

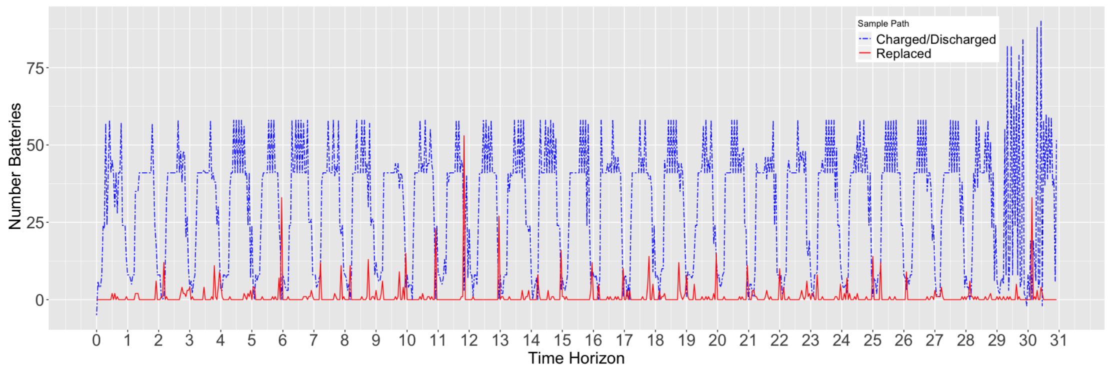


Figure 3.7: Sample path of the policy for scenario 15 when realized demand equals mean demand.

3.7 Conclusions

We examined a stochastic scheduling, allocation, and inventory replenishment problem (SAIRP) for a battery swap station where there is a direct link between the inventory level and necessary recharging and replacement actions for battery charge and battery capacity. Specifically, the di-

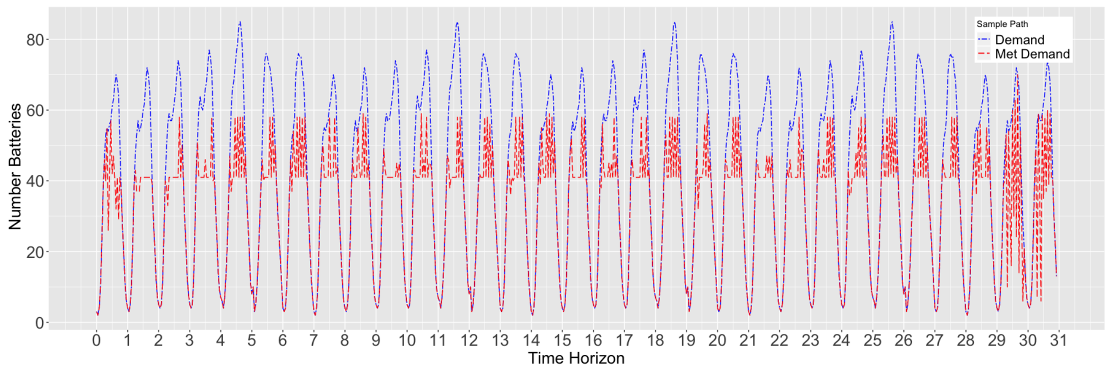


Figure 3.8: Demand and met demand for scenario 15 based on the sample path when realized demand equals mean demand.

rect link is that the act of recharging a battery to enable short-term operation is the exact cause for long-term battery capacity degradation. This creates a unique problem where trade-offs between recharging and replacing batteries must be analyzed. We utilized a Markov Decision Process (MDP) model for a battery swap station faced with battery degradation and uncertain arrival of demand for swaps. In the MDP model of the stochastic SAIRP, we determine the optimal policy for charging, discharging, and replacing batteries over time when faced with non-stationary charging prices, non-stationary discharging revenue, and capacity-dependent swap revenue. We prove theoretical properties for the MDP model, including the existence of an optimal monotone policy for the second dimension of the state when there is an upper bound placed on the number of batteries replaced in each decision epoch. Further, we prove the monotonicity of the value function. Given these results, we solved the stochastic SAIRP using both backward induction (BI) and monotone backward induction (MBI). However, we run into the curses of dimensionality as we are unable to find an optimal policy for realistic-sized instances.

To overcome these curses, we propose a monotone approximate dynamic programming (ADP) solution method with regression-based initialization (MADP-RB). The MADP-RB builds upon the monotone ADP (MADP) algorithm, which is fitting for problems with monotone value functions (Jiang and Powell, 2015). In our MADP-RB, we initialize the value function based on an intelligent approximation using regression. We used a Latin hypercube designed experiment to test the performance of MBI, MADP, and MADP-RB. In the designed experiment, we examine both modest-sized problem instances that are optimally solvable using BI and realistic-sized instances. Overall, we observed that MADP-RB resulted in the greatest performance in terms of computational time, optimality gap, convergence, and expected total reward.

Our investigation into SAIRPs opens the avenue for many opportunities for future work. In terms of the model and solution method, researchers should examine state reduction/aggregation approaches and quantify how they impact the quality of the solutions and computational time. Future research should consider different mechanisms within a disaggregated MDP model that captures individual battery states and actions. Furthermore, it is valuable to solve the disaggregated MDP using approximate solution methods and compare the insights with the presented aggregated MDP. Moreover, it is interesting to study discretizing the state of charge of batteries and allow partial recharging decisions. For the battery swap station application, future work should consider different types of charging options, multiple swap station locations, and multiple classes of demand dictated by how long the battery must operate to satisfy the demand (e.g., how far a drone must fly).

Acknowledgement

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.

Bibliography

Abdollahi, A., Raghunathan, N., Han, X., Pattipati, B., Balasingam, B., Pattipati, K. R., Bar-Shalom, Y., and Card, B. (2015). Battery health degradation and optimal life management. In *IEEE AUTOTESTCON*, pages 146–151, National Harbor, MD, USA.

Abe, M., Nishimura, K., Seki, E., Haruna, H., Hirasawa, T., Ito, S., and Yoshiura, T. (2012). Lifetime prediction for heavy-duty industrial lithium-ion batteries that enables highly reliable system design. *Hitachi Review*, 61(6):259–263.

Alagoz, O., Maillart, L. M., Schaefer, A. J., and Roberts, M. S. (2004). The optimal timing of living-donor liver transplantation. *Management Science*, 50(10):1420–1430.

Battery University (2017). BU-801b: How to define battery life. Last accessed on April 1, 2021 at: https://batteryuniversity.com/learn/article/how_to_define_battery_life#:~:text=The%20service%20life%20of%20a,elevated%20temperatures%20also%20induces%20stress.

Bertsimas, D. and Demir, R. (2002). An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science*, 48(4):550–565.

Bloch-Mercier, S. (2001). Monotone markov processes with respect to the reversed hazard rate ordering: an application to reliability. *Journal of Applied Probability*, 38(1):195–208.

CBS NEWS (2013). Amazon unveils futuristic plan: Delivery by drone. Last accessed on July 12, 2020 at http://www.cbsnews.com/news/amazon-unveils-futuristic-plan-delivery-by-drone/.

Chhatwal, J., Alagoz, O., and Burnside, E. S. (2010). Optimal breast biopsy decision-making based on mammographic features and demographic factors. *Operations Research*, 58(6):1577–1591.

Çimen, M. and Kirkbride, C. (2013). Approximate dynamic programming algorithms for multidimensional inventory optimization problems. *IFAC Proceedings Volumes*, 46(9):2015–2020.

Çimen, M. and Kirkbride, C. (2017). Approximate dynamic programming algorithms for multidimensional flexible production-inventory problems. *International Journal of Production Research*, 55(7):2034–2050.

Clark, A. J. (1972). An informal survey of multi-echelon inventory theory. *Naval Research Logistics*, 19(4):621–650.

Clark, A. J. and Scarf, H. (1960). Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490.

Dai, Q., Cai, T., Duan, S., and Zhao, F. (2014). Stochastic modeling and forecasting of load demand for electric bus battery-swap station. *IEEE Transactions on Power Delivery*, 29(4):1909–1917.

Darken, C. and Moody, J. (1992). Towards faster stochastic gradient search. In *Neural Information Processing Systems*, volume 4, pages 1009–1016, San Mateo, CA. Morgan Kaufmann.

de Jonge, B. and Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, 285(3):805–824.

Debnath, U. K., Ahmad, I., and Habibi, D. (2014). Quantifying economic benefits of second life batteries of gridable vehicles in the smart grid. *International Journal of Electrical Power and Energy Systems*, 63:577–587.

DeCroix, G. A. and Arreola-Risa, A. (1998). Optimal production and inventory policy for multiple products under resource constraints. *Management Science*, 44(7):950–961.

DHL Press Release (2016). Successful Trial Integration of DHL Parcelcopter into Logistics Chain. Last accessed on June 7, 2021 at http://www.dhl.com/en/press/releases/releases_2016/all/parcel_ecommerce/successful_trial_integration_dhl_parcelcopter_logistics_chain.html.

DJI (2019). DJI Spreading Wings S1000 specs. Last accessed on June 7, 2021 at https://www.dji.com/spreading-wings-s1000/spec.

Drones Etc. (2015). Phantom 3 batteries – Tips and Tutorial. Last accessed on June 7, 2021 at https://www.dronesetc.com/blogs/news/30238721-phantom-3-batteries-tips-and-tutorial.

Dubarry, M., Truchot, C., Liaw, B. Y., Gering, K., Sazhin, S., Jamison, D., and Michelbacher, C. (2011). Evaluation of commercial lithium-ion cells based on composite positive electrode for plug-in hybrid electric vehicle applications. part II. degradation mechanism under 2C cycle aging. *Journal of Power Sources*, 196(23):10336–10343.

Dunn, B., Kamath, H., and Tarascon, J.-M. (2011). Electrical energy storage for the grid: A battery of choices. *Science*, 334(6058):928–935.

ElHafsi, M. (2009). Optimal integrated production and inventory control of an assemble-to-order system with multiple non-unitary demand classes. *European Journal of Operational Research*, 194(1):127–142.

Elwany, A. H. and Gebraeel, N. Z. (2008). Sensor-driven prognostic models for equipment replacement and spare parts inventory. *IIE Transactions*, 40(7):629–639.

Erdelyi, A. and Topaloglu, H. (2010). Approximate dynamic programming for dynamic capacity allocation with multiple priority levels. *IIE Transactions*, 43(2):129–142.

Gao, Y., Zhao, K., and Wang, C. (2012). Economic dispatch containing wind power and electric vehicle battery swap station. In *IEEE PES Transmission and Distribution Conference and Exposition*, pages 1–7, Orlando, FL.

George, A. P. and Powell, W. B. (2006). Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65(1):167–198.

Golari, M., Fan, N., and Jin, T. (2017). Multistage stochastic optimization for production-inventory planning with intermittent renewable energy. *Production and Operations Management*, 26(3):409–425.

Govindan, K., Soleimani, H., and Kannan, D. (2015). Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future. *European Journal of Operational Research*, 240(3):603–626.

Hong, I., Kuby, M., and Murray, A. T. (2018). A range-restricted recharging station coverage model for drone delivery service planning. *Transportation Research Part C: Emerging Technologies*, 90:198–212.

Horenbeek, A. V., Buré, J., Cattrysse, D., Pintelon, L., and Vansteenwegen, P. (2013). Joint maintenance and inventory optimization systems: A review. *International Journal of Production Economics*, 143(2):499–508.

Hussein, A. (2015). Capacity fade estimation in electric vehicle li-ion batteries using artificial neural networks. *IEEE Transactions on Industry Applications*, 51(3):2321–2330.

Inside EVs (2020). E.ON and Nissan are launching new V2G trial project in the UK. Last accessed on March 24, 2021 at https://insideevs.com/news/437785/eon-nissan-v2g-trial-project-uk/.

James, D. (2020). 14 drones with the best flight times. Last accessed on June 7, 2021 at http://www.dronesglobe.com/guide/long-flight-time/.

Jensen, J. (2019). Agricultural drones: How drones are revolutionizing agriculture and how to break into this booming market. UAV Coach. Last accessed on May 21, 2021 at: https://uavcoach.com/agricultural-drones/.

Jiang, D. R. and Powell, W. B. (2015). An approximate dynamic programming algorithm for monotone value functions. *Operations Research*, 63(6):1489–1511.

Kang, Q., Wang, J., Zhou, M., and Ammari, A. C. (2016). Centralized charging strategy and scheduling algorithm for electric vehicles under a battery swapping scenario. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):659–669.

Khojandi, A., Maillart, L. M., Prokopyev, O. A., Roberts, M. S., Brown, T., and Barrington, W. W. (2014). Optimal implantable cardioverter defibrillator (ICD) generator replacement. *INFORMS Journal on Computing*, 26(3):599–615.

Kim, J., Song, B. D., and Morrison, J. R. (2013). On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems*, 70(1):347–359.

Kwizera, O. and Nurre, S. G. (2018). Using drones for delivery: A two-level integrated inventory problem with battery degradation and swap stations. In *Proceedings of the Industrial and Systems Engineering Research Conferences*, pages 1–6, Orlando, FL.

Laan, E. V. D. and Salomon, M. (1997). Production planning and inventory control with remanufacturing and disposal. *European Journal of Operational Research*, 102(2):264–278.

Lacey, G., Jiang, T., Putrus, G., and Kotter, R. (2013). The effect of cycling on the state of health of the electric vehicle battery. In *48th International Universities' Power Engineering Conference*, pages 1–7, Dublin, Ireland.

Maity, A. K. (2011). One machine multiple-product problem with production-inventory system under fuzzy inequality constraint. *Applied Soft Computing*, 11(2):1549–1555.

Mak, H.-Y., Rong, Y., and Shen, Z.-J. M. (2013). Infrastructure planning for electric vehicles with battery swapping. *Management Science*, 59(7):1557–1575.

Maxwell, M. S., Restrepo, M., Henderson, S. G., and Topaloglu, H. (2010). Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281.

Meissner, J. and Senicheva, O. V. (2018). Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems. *European Journal of Operational Research*, 265(1):49–64.

Montgomery, D. (2008). *Design and analysis of experiments*. John Wiley & Sons, New Jersey, 8th edition.

Morrow, K., Karner, D., and Francfort, J. (2008). Plug-in hybrid electric vehicle charging infrastructure review. Technical report, U.S. Department of Energy, Idaho National Laboratory.

Mutzabaugh, B. (2017). Drone taxis? Dubai plans roll out of self-flying pods. Last accessed on May 21, 2021 at https://www.usatoday.com/story/travel/flights/todayinthesky/2017/02/13/dubai-passenger-carrying-drones-could-flying-july/97850596/.

Nahmias, S. (1982). Perishable inventory theory: A review. *Operations Research*, 30(4):680–708.

Nasrollahzadeh, A., Khademi, A., and Mayorga, M. E. (2018). Real-time ambulance dispatching and relocation. *Manufacturing and Service Operations Management*, 20(3):467–480.

National Grid (2016). Hourly electric supply charges. Last accessed on June 7, 2021 at https://www.nationalgridus.com/niagaramohawk/business/rates/5_hour_charge.asp.

Nexant, Inc., Air Liquide, Argonne National Laboratory, Chevron Technology Venture, Gas Technology Institute, National Renewable Energy Laboratory, Pacific Northwest National Laboratory, and TIAX LLC (2008). H2A hydrogen delivery infrastructure analysis models and conventional pathway options analysis results. Online. Last accessed on June 7, 2021 at https://www.energy.gov/sites/prod/files/2014/03/f9/nexant_h2a.pdf.

Nurre, S. G., Bent, R., Pan, F., and Sharkey, T. C. (2014). Managing operations of plug-in hybrid electric vehicle (PHEV) exchange stations for use with a smart grid. *Energy Policy*, 67:364–377.

Nuvve (2021a). Toyota Tsusho and Chubu Electric Power announce Japan's first ever V2G project using Nuvve's technology. Last accessed on March 24, 2021 at https://nuvve.com/toyota-tsusho-and-chubu-electric-power-announce-japans-first-ever-v2g-project-using-nuvves-technology/.

Nuvve (2021b). Vehicle-To-Grid technology. Last accessed on March 24, 2021 at https://nuvve.com/technology/.

Pan, F., Bent, R., Berscheid, A., and Izraelevitz, D. (2010). Locating PHEV exchange stations in V2G. In *First IEEE International Conference on Smart Grid Communications*, pages 173–178, Gaithersburg, MD, USA.

Papadakia, K. and Powell, W. B. (2007). Monotonicity in multidimensional Markov decision processes for the batch dispatch problem. *Operations Research Letters*, 35(2):267–272.

Park, S., Zhang, L., and Chakraborty, S. (2017). Battery assignment and scheduling for drone delivery businesses. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 1–6, Taipei, Taiwan.

Plett, G. L. (2011). Recursive approximate weighted total least squares estimation of battery cell total capacity. *Journal of Power Sources*, 196(4):2319–2331.

Porteus, E. L. (2002). *Foundations of Stochastic Inventory Theory*. Stanford University Press, Stanford, CA.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, New York, NY, USA, 2 edition.

Powell, W. B. and Topaloglu, H. (2005). Approximate dynamic programming for large-scale resource allocation problems. *INFORMS TutORials in Operations Research*, pages 123–147.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Hoboken, New Jersey, 1st edition.

Rausch, M. and Liao, H. (2010). Joint production and spare part inventory control strategy driven by condition based maintenance. *IEEE Transactions on Reliability*, 59(3):507–516.

Rettke, A. J., Robbins, M. J., and Lunday, B. J. (2016). Approximate dynamic programming for the dispatch of military medical evacuation assets. *European Journal of Operational Research*, 254(3):824–839.

Rezvani, Z., Jansson, J., and Bodin, J. (2015). Advances in consumer electric vehicle adoption research: A review and research agenda. *Transportation Research Part D: Transport and Environment*, 34:122–136.

Ribbernick, H., Darcovich, K., and Pincet, F. (2015). Battery life impact of vehicle-to-grid application of electric vehicles. In *28th International Electric Vehicle Symposium and Exhibition*, volume 2, pages 1535–1545, Goyang, Korea. Korean Society of Automotive Engineers.

Romm, J. (2017). Chart of the month: Driven by Tesla, battery prices cut in half since 2014. Last accessed on June 7, 2021 at https://archive.thinkprogress.org/chart-of-the-month-driven-by-tesla-battery-prices-cut-in-half-since-2014-718752a30a42/.

Sarker, M., Pandžić, H., and Ortega-Vazquez, M. (2015). Optimal operation and services scheduling for an electric vehicle battery swapping station. *IEEE Transactions on Power Systems*, 30(2):901–910.

Saxena, S., Floch, C. L., MacDonald, J., and Moura, S. (2015). Quantifying EV battery end-of-life through analysis of travel needs with vehicle powertrain models. *Journal of Power Sources*, 282:265–276.

Scarf, H. (1960). The optimality of (S,s) policies in the dynamic inventory problem. In Arrow, Karlin, and Suppers, editors, *Mathematical Methods in the Social Sciences*, pages 196–202, Stanford, CA. Stanford University Press.

Schneider, F., Thonemann, U. W., and Klabjan, D. (2018). Optimization of battery charging and purchasing at electric vehicle battery swap stations. *Transportation Science*, 52(5):1211–1234.

Shavarani, S. M., Nejad, M. G., Rismanchian, F., and Izbirak, G. (2018). Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco. *The International Journal of Advanced Manufacturing Technology*, 95(9):3141–3153.

Shen, Z.-J. M., Feng, B., Mao, C., and Ran, L. (2019). Optimization models for electric vehicle service operations: A literature review. *Transportation Research Part B: Methodological*, 128:462–477.

Shirk, M. and Wishart, J. (2015). Effects of electric vehicle fast charging on battery life and vehicle performance. In *SAE Technical Paper*, pages 1–13, Detroit, MI. SAE 2015 World Congress & Exhibition, SAE International.

Simao, H. P., Day, J., George, A. P., Gifford, T., Nienow, J., and Powell, W. B. (2009). An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197.

Sun, B., Tan, X., and Tsang, D. H. K. (2014). Optimal charging operation of battery swapping stations with QoS guarantee. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 13–18, Venice, Italy.

Tan, X., Qu, G., Sun, B., Li, N., and Tsang, D. H. K. (2019). Optimal scheduling of battery charging station serving electric vehicles based on battery swapping. *IEEE Transactions on Smart Grid*, 10(2):1372–1384.

Tesla (2017). Supercharger. Last accessed on June 7, 2021 at https://www.tesla.com/supercharger.

Toktay, L. B., Wein, L. M., and Zenios, S. A. (2000). Inventory management of remanufacturable products. *Management Science*, 46(11):1412–1426.

United States Department of Energy (2014). EV everywhere grand challenge road to success. Last accessed on June 7, 2021 at https://www.energy.gov/sites/prod/files/2014/02/f8/eveverywhere_road_to_success.pdf.

Wallace, T. (2015). Royal Mail wants to use drones and driverless trucks. Last accessed on June 7, 2021 at The Telegraph: http://www.telegraph.co.uk/technology/11984099/Royal-Mail-wants-to-use-drones-and-driverless-trucks.html.

Weise, E. (2017). UPS tested launching a drone from a truck for deliveries. USA Today. Last accessed on May 21, 2021 at https://www.usatoday.com/story/tech/news/2017/02/21/ups-delivery-top-of-van-drone-workhorse/98057076/.

Widrick, R. S., Nurre, S. G., and Robbins, M. J. (2018). Optimal policies for the management of an electric vehicle battery swap station. *Transportation Science*, 52(1):59–79.

Wood, E., Alexander, M., and Bradley, T. H. (2011). Investigation of battery end-of-life conditions for plug-in hybrid electric vehicles. *Journal of Power Sources*, 196(11):5147–5154.

Worley, O. and Klabjan, D. (2011). Optimization of battery charging and purchasing at Electric Vehicle battery swap stations. In *IEEE Vehicle Power and Propulsion Conference*, pages 1–4, Chicago, IL.

Xu, B., Oudalov, A., Ulbig, A., Andersson, G., and Kirschen, D. S. (2018). Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140.

Yang, J. and Sun, H. (2015). Battery swap station location-routing problem with capacitated electric vehicles. *Computers and Operations Research*, 55:217–232.

Zhang, J., Denton, B. T., Balasubramanian, H., Shah, N. D., and Inman, B. A. (2012). Optimization of prostate biopsy referral decisions. *Manufacturing & Service Operations Management*, 14(4):529–547.

Zhang, L., Lou, S., Wu, Y., Yi, L., and Hu, B. (2014). Optimal scheduling of electric vehicle battery swap station based on time-of-use pricing. In *IEEE PES Asia-Pacific Power and Energy Engineering Conference*, pages 1–6, Hong Kong.

Zhang, P. (2020). NIO signs agreement with state grid subsidiary to build 100 charging and battery swap stations by 2021. Last accessed on March 23, 2021 at https://cntechpost.com/2020/12/14/nio-national-grid-subsidiary-to-build-100-charging-battery-swap-stations-by-2021/.

Zhou, S. X., Tao, Z., and Chao, X. (2011). Optimal control of inventory systems with multiple types of remanufacturable products. *Manufacturing & Service Operations Management*, 13(1):20–34.

Appendix

3.A Monotonicity of the Second Dimension of the State

In this Appendix, in Lemma 1, we prove that the stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state. Then, in Theorem 1, we prove the special conditions under which a monotone policy in the second dimension of the state is optimal. First, we state Lemma 1 and then prove Lemmas 2, 3, 4, and 5 that we use to prove Lemma 1 and Theorem 1.

Lemma 1. *The stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state.*

Lemma 2. *If $s_t^2 \geq \widetilde{s_t^2}$ and a_t^2, a_t^1, s_t^1 are constant then $j^2 \geq \widetilde{j^2}$.*

Proof. Using Equation (3.4), the second state of the system transitions from s_t^2 to j^2 and from $\widetilde{s_t^2}$ to $\widetilde{j^2}$ when action $a_t = (a_t^1, a_t^2)$ is taken. In Equation (3.4), we use the *round*() function to return values in the discretized state space. Without loss of generality, we examine the state transition for the second state of the system without the *round*() function, as the *round*() function does not alter the validity of the arguments. Thus, we have

$$\begin{aligned} j^2 = s_{t+1}^2 &= \frac{1}{M}\left[s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})\right], \\ \widetilde{j^2} = \widetilde{s_{t+1}^2} &= \frac{1}{M}\left[\widetilde{s_t^2}(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})\right]. \end{aligned} \tag{3.24}$$

We start from $j^2 \geq \widetilde{j^2}$ and reduce it to an always true statement.

$$\begin{gathered} j^2 \geq \widetilde{j^2} \Leftrightarrow \\ \tfrac{1}{M}\left[s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})\right] \geq \tfrac{1}{M}\left[\widetilde{s_t^2}(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})\right] \Leftrightarrow \\ s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-}) \geq \widetilde{s_t^2}(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-}) \Leftrightarrow \\ s_t^2(M-a_t^2) \geq \widetilde{s_t^2}(M-a_t^2). \end{gathered}$$

As $a_t^2 \leq M$, $(M-a_t^2)$ is always non-negative. Thus, we can divide both sides of the last inequality by this non-negative value and get $s_t^2 \geq \widetilde{s_t^2}$ which is always true.

□

Lemma 3. *If s_t^2, a_t^1, and θ are fixed, then replacing less than*

$$\xi = \left\lfloor \frac{\delta^C(a_t^{1+}+a_t^{1-})-M(s_t^2-\theta)}{1-s_t^2} \right\rfloor$$

causes a transition to absorbing state η.

Proof. Transition from state s_t^2 to absorbing state η means $s_{t+1}^2 < \theta$. From Equation (3.24) we know that

$$s_{t+1}^2 = \frac{1}{M}\left[s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})\right].$$

Therefore,

$$\begin{aligned}
s_{t+1}^2 = j^2 < \theta &\Leftrightarrow \\
\frac{1}{M}[s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-})] < \theta &\Leftrightarrow \\
s_t^2(M-a_t^2)+a_t^2-\delta^C(a_t^{1+}+a_t^{1-}) < M\theta &\Leftrightarrow \\
(1-s_t^2)a_t^2 < M\theta+\delta^C(a_t^{1+}+a_t^{1-})-Ms_t^2 &\Leftrightarrow \\
a_t^2 < \frac{M\theta+\delta^C(a_t^{1+}+a_t^{1-})-Ms_t^2}{1-s_t^2} &\Leftrightarrow \\
a_t^2 < \frac{\delta^C(a_t^{1+}+a_t^{1-})-M(s_t^2-\theta)}{1-s_t^2}&.
\end{aligned}$$

Because a_t^2 is an integer, the upper bound can be derived using the floor function.

$$a_t^2 < \xi = \left\lfloor \frac{\delta^C(a_t^{1+}+a_t^{1-})-M(s_t^2-\theta)}{1-s_t^2} \right\rfloor.$$

□

In this Appendix, when we use the transition probability and reward functions with only the second dimension of the state and action as arguments, we assume that the first dimension of the system is fixed (i.e., s_t^1 and a_t^1 are constants).

Lemma 4. *For the stochastic SAIRP, $q_t(k \mid s_t^2, a_t^2)$ is a deterministic function equal to*

$$q_t(k \mid s_t^2, a_t^2) = \begin{cases} 0 & \text{if} \quad j^2 < k, \\ p_t(j^2 \mid s_t^2, a_t^2) = 1 & \text{if} \quad j^2 \geq k. \end{cases}$$

Proof. As defined in Section 3.3,

$$q_t(k \mid s, a) = \sum_{j=k}^{\infty} p_t(j \mid s, a).$$

Hence, when the first dimension of the system is fixed, we have

$$q_t(k \mid s_t^2, a_t^2) = \sum_{j^2=k}^{\infty} p_t(j^2 \mid s_t^2, a_t^2).$$

From the second state transition function given by Equation (3.5), the value of second state in the future, $s_{t+1}^2 = j^2$, does not depend on D_t. From Lemma 3, if we start from s_t^2 and take action a_t^2, the capacity will transition to the absorbing state $j^2 = \eta$, if

$$a_t^2 < \lfloor \xi \rfloor$$

or it will transition to state $j^2 \geq \theta$, if

$$a_t^2 \geq \left\lfloor \xi \right\rfloor.$$

Thus, this is a deterministic transition with into two intervals, $\left[\eta, j^2\right]$ and $(j^2, 1\big]$. If $k \in (j^2, 1\big]$, then $q_t(k \mid s_t^2, a_t^2)$ equals zero. However, if $k \in \left[\eta, j^2\right]$, then $k \leq j^2$ and $q_t(k \mid s_t^2, a_t^2)$ equals 1.

□

We use the notation we define in Definition 1 in Lemma 5, Lemma 1, and Theorem 1.

Definition 1. *The second dimension of the state transitions to the following state when starting from s_t^2 and taking action a_t^2.*

$$j_A^2 = \frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right]. \tag{3.25}$$

The second dimension of the state transitions to the following state when starting from $\widetilde{s_t^2}$ and taking action a_t^2.

$$j_B^2 = \frac{1}{M}\left[\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right]. \tag{3.26}$$

The second dimension of the state transitions to the following state when starting from s_t^2 and taking action $\widetilde{a_t^2}$.

$$j_C^2 = \frac{1}{M}\left[s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right]. \tag{3.27}$$

The second dimension of the state transitions to the following state when starting from $\widetilde{s_t^2}$ and taking action $\widetilde{a_t^2}$.

$$j_D^2 = \frac{1}{M}\left[\widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right]. \tag{3.28}$$

Lemma 5. *If $s_t^2 \geq \widetilde{s_t^2}$ and $a_t^2 \geq \widetilde{a_t^2}$, then $j_A^2 \geq \max(j_B^2, j_C^2) \geq \min(j_B^2, j_C^2) \geq j_D^2$.*

Proof. Based on Definition 1, we need to show j_A^2 has the greatest value and j_D^2 has the lowest value. Thus, as follows we prove $j_A^2 \geq j_B^2$, $j_A^2 \geq j_C^2$, $j_A^2 \geq j_D^2$, $j_B^2 \geq j_D^2$, and $j_C^2 \geq j_D^2$. We proceed with a proof by cases starting with the claim and reducing it to an always true statement.

i. $j_A^2 \geq j_D^2$

$$j_A^2 \geq j_D^2 \Leftrightarrow$$

$$\frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[\widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$

$$s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$

$$Ms_t^2 - s_t^2 a_t^2 + a_t^2 - M\widetilde{s_t^2} + \widetilde{s_t^2}\widetilde{a_t^2} - \widetilde{a_t^2} \geq 0 \Leftrightarrow$$

$$M(s_t^2 - \widetilde{s_t^2}) + a_t^2(1 - s_t^2) - \widetilde{a_t^2}(1 - \widetilde{s_t^2}) \geq 0.$$

We know $a_t^2(1 - s_t^2) \geq \widetilde{a_t^2}(1 - s_t^2)$, so it suffices to show that

$$M(s_t^2 - \widetilde{s_t^2}) + \widetilde{a_t^2}(1 - s_t^2) - \widetilde{a_t^2}(1 - \widetilde{s_t^2}) \geq 0 \Leftrightarrow$$

$$M(s_t^2 - \widetilde{s_t^2}) + \widetilde{a_t^2}(\widetilde{s_t^2} - s_t^2) \geq 0 \Leftrightarrow$$
$$(M - \widetilde{a_t^2})(s_t^2 - \widetilde{s_t^2}) \geq 0.$$

Because $M \geq \widetilde{a_t^2}$ and $s_t^2 \geq \widetilde{s_t^2}$ the last statement is always non-negative and we prove our claim that $j_A^2 \geq j_D^2$.

ii. $j_A^2 \geq j_B^2$

$$j_A^2 \geq j_B^2 \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$
$$s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \geq \widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$
$$s_t^2(M - a_t^2) \geq \widetilde{s_t^2}(M - a_t^2).$$

Because $M \geq \widetilde{a_t^2}$ we have
$$s_t^2 \geq \widetilde{s_t^2}.$$

The last statement is always true and we prove our claim that $j_A^2 \geq j_B^2$.

iii. $j_A^2 \geq j_C^2$

$$j_A^2 \geq j_C^2 \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$
$$s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \geq s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$
$$s_t^2(M - a_t^2) + a_t^2 \geq s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} \Leftrightarrow$$
$$s_t^2(M - a_t^2 - M + \widetilde{a_t^2}) + a_t^2 - \widetilde{a_t^2} \geq 0 \Leftrightarrow$$
$$s_t^2(\widetilde{a_t^2} - a_t^2) + a_t^2 - \widetilde{a_t^2} \geq 0 \Leftrightarrow$$
$$(1 - s_t^2)(a_t^2 - \widetilde{a_t^2}) \geq 0.$$

Because $(1 - s_t^2) \geq 0$ and $(a_t^2 - \widetilde{a_t^2}) \geq 0$ the last statement is always non-negative and we prove our claim that $j_A^2 \geq j_C^2$.
So far we know $j_A^2 \geq \max(j_B^2, j_C^2) \geq \min(j_B^2, j_C^2)$ and $j_A^2 \geq j_D^2$. Now, we show $\min(j_B^2, j_C^2) \geq j_D^2$. It suffices to show $j_B^2 \geq j_D^2$ and $j_C^2 \geq j_D^2$. First, we show $j_B^2 \geq j_D^2$.

iv. $j_B^2 \geq j_D^2$

$$j_B^2 \geq j_D^2 \Leftrightarrow$$
$$\frac{1}{M}\left[\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[\widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$
$$\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$
$$\widetilde{s_t^2}(M - a_t^2) + a_t^2 \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} \Leftrightarrow$$

$$\widetilde{s_t^2}(\widetilde{a_t^2} - a_t^2) + (a_t^2 - \widetilde{a_t^2}) \geq 0 \Leftrightarrow$$
$$(1 - \widetilde{s_t^2})(a_t^2 - \widetilde{a_t^2}) \geq 0.$$

Because $(1 - \widetilde{s_t^2}) \geq 0$ and $(a_t^2 - \widetilde{a_t^2}) \geq 0$ the last statement is always non-negative and we prove our claim that $j_B^2 \geq j_D^2$.

v. $j_C^2 \geq j_D^2$

$$j_C^2 \geq j_D^2 \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[\widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$
$$s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-}) \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$
$$s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} \Leftrightarrow$$
$$s_t^2(M - \widetilde{a_t^2}) \geq \widetilde{s_t^2}(M - \widetilde{a_t^2}) \Leftrightarrow$$
$$s_t^2 \geq \widetilde{s_t^2}.$$

The last statement is always true and we prove our claim that $j_C^2 \geq j_D^2$. From iv and v, we conclude that $\min(j_B^2, j_C^2) \geq j_D^2$. From i, ii, iii, iv, and v, we prove
$$j_A^2 \geq \max(j_B^2, j_C^2) \geq \min(j_B^2, j_C^2) \geq j_D^2.$$

$\square$

Using Lemmas 2, 3, 4, and 5, we prove in Lemma 1 that the stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state. Then, in Theorem 1, we prove the special conditions under which a monotone policy is optimal in the second dimension of the state.

Lemma 1. *The stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state.*

Proof. First, we need to show that at least one of the following conditions is not always satisfied for our problem. We will prove that condition 4 is not satisfied; That is, $q_t(k \mid s_t^2, a_t^2)$ **is not** subadditive on $S^2 \times A'$ for every $k \in S^2$ where A' is the set of possible actions in the second dimension, independent of the state of the system.

1. $r_t(s_t^2, a_t^2)$ is non-decreasing in s_t^2 for all $a_t^2 \in A'$.
   If $s_t^2 \geq \widetilde{s_t^2}$, it suffices to show that $r_t(s_t^2, a_t^2) \geq r_t^2(\widetilde{s_t^2}, a_t^2)$. From Equations (3.7) and (3.8) we know

$$r_t(s_t^2, a_t^2) = \frac{\beta(1 + s_t^2 - 2\theta)}{1 - \theta}\left[\min\{D_t, s_t^1 - a_t^{1-}\}\right] - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2.$$

We start with our claim, $r_t(s_t^2, a_t^2) \geq r_t(\widetilde{s_t^2}, a_t^2)$, and reduce it to an always true statement.

$$r_t(s_t^2, a_t^2) \geq r_t(\widetilde{s_t^2}, a_t^2) \Leftrightarrow$$

$$\tfrac{\beta(1+s_t^2-2\theta)}{1-\theta}\left[\min\{D_t, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_ta_t^2 \geq$$
$$\tfrac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta}\left[\min\{D_t, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_ta_t^2 \Leftrightarrow$$
$$\tfrac{\beta(1+s_t^2-2\theta)}{1-\theta} \geq \tfrac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta} \Leftrightarrow$$
$$s_t^2 \geq \widetilde{s_t^2}.$$

2. $q_t(k \mid s_t^2, a_t^2)$ is non-decreasing in s_t^2 for all $k \in S^2$ and $a \in A'$.
If $s_t^2 \geq \widetilde{s_t^2}$, we seek to show

$$q_t(k \mid s_t^2, a_t^2) \geq q_t(k \mid \widetilde{s_t^2}, a_t^2).$$

It is sufficient to show that if $j_A^2 \geq j_B^2$ from Definition 1, then $q_t(k \mid s_t^2, a_t^2) \geq q_t(k \mid \widetilde{s_t^2}, a_t^2)$. We show $j_A^2 \geq j_B^2$ in Lemma 2 and prove our claim.

3. $r_t(s_t^2, a_t^2)$ is a subadditive function on $S^2 \times A'$.
For $r_t(s_t^2, a_t^2)$ to be subadditive, it means the incremental effect on the expected total reward of replacing less batteries is less when the average capacity is greater. Consider, $s_t^2 \geq \widetilde{s_t^2}$ and $a_t^2 \geq \widetilde{a_t^2}$. It suffices to show that

$$r_t(s_t^2, a_t^2) + r_t(\widetilde{s_t^2}, \widetilde{a_t^2}) \leq r_t(s_t^2, \widetilde{a_t^2}) + r_t(\widetilde{s_t^2}, a_t^2) \Leftrightarrow$$

$$\sum_{j=0}^{\infty} P(D_t = j)\frac{\beta(1+s_t^2-2\theta)}{1-\theta}\left[\min\{j, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_ta_t^2$$
$$+\sum_{j=0}^{\infty} P(D_t = j)\frac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta}\left[\min\{j, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_t\widetilde{a_t^2} \leq$$
$$\sum_{j=0}^{\infty} P(D_t = j)\frac{\beta(1+s_t^2-2\theta)}{1-\theta}\left[\min\{j, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_t\widetilde{a_t^2}$$
$$+\sum_{j=0}^{\infty} P(D_t = j)\frac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta}\left[\min\{j, s_t^1-a_t^{1-}\}\right]-K_ta_t^{1+}+J_ta_t^{1-}-L_ta_t^2.$$

As the right-hand side and the left-hand side of the inequality are the same, we prove our claim.

4. $q_t(k \mid s_t^2, a_t^2)$ is subadditive on $S^2 \times A'$ for every $k \in S^2$. It suffices to show that

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}),$$

for every $k \in S^2$ if $s_t^2 \geq \widetilde{s_t^2}$ and $a_t^2 \geq \widetilde{a_t^2}$.

From Lemma 4,

$$q_t(k \mid s_t^2, a_t^2) = \begin{cases} 0 & \text{if} \quad j^2 < k, \\ p_t(j^2 \mid s_t^2, a_t^2) = 1 & \text{if} \quad j^2 \geq k. \end{cases}$$

From Lemmas 3 and 4, the starting state s_t^2 transfers to j_A^2 or j_C^2, if we take action a_t^2 or $\widetilde{a_t^2}$, respectively. If the number of batteries replaced is greater than a threshold ξ, then the future state is at least the capacity threshold, θ. Otherwise, we will transfer to the absorbing state, η. Similarly, the starting state $\widetilde{s_t^2}$ transfers to j_B^2 or j_D^2, if we take action a_t^2 or $\widetilde{a_t^2}$, respectively. If the number of batteries replaced is greater than a threshold

$$\widetilde{\xi} = \left\lfloor \frac{\delta^C(a_t^{1+} + a_t^{1-}) - M(\widetilde{s_t^2} - \theta)}{1 - \widetilde{s_t^2}} \right\rfloor,$$

then the future state is at least θ. Otherwise, we will transfer to the absorbing state, η. Recall in Lemma 5 and Definition 1 we have:

$$j_A^2 = \frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right],$$

$$j_B^2 = \frac{1}{M}\left[\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right],$$

$$j_C^2 = \frac{1}{M}\left[s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right],$$

$$j_D^2 = \frac{1}{M}\left[\widetilde{s_t^2}(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right],$$

and

$$j_A^2 \geq \max(j_B^2, j_C^2) \geq \min(j_B^2, j_C^2) \geq j_D^2.$$

To show the subadditivity property of the $q_t(k \mid s_t^2, a_t^2)$ for every $k \in S^2$, we divide S^2 space into five intervals. Then, we show that if $k \in \left(\max(j_B^2, j_C^2), j_A^2\right]$ the subadditivity condition is not satisfied.

i. $0 \leq k \leq j_D^2$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$

$$1 + 1 \leq 1 + 1.$$

ii. $j_D^2 < k \leq \min(j_B^2, j_C^2)$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$

$$1 + 0 \leq 1 + 1.$$

iii. $\min(j_B^2, j_C^2) < k \leq \max(j_B^2, j_C^2)$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$

$$1 + 0 \leq 1 + 0.$$

iv. $\max(j_B^2, j_C^2) < k \leq j_A^2$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$

$$1 + 0 \nleq 0 + 0.$$

v. $j_A^2 < k$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$

$$0 + 0 \leq 0 + 0.$$

Due to the result of part iv, we **can not** conclude that $q_t(k \mid s_t^2, a_t^2)$ is subadditive on $S^2 \times A'$ for every $k \in S^2$.

5. $r_N(s_N)$ is non-decreasing in s_N^2.
Consider $s_N^2 \geq \widetilde{s_N^2}$, it suffices to show that $r_N(s_N) \geq r_N(\widetilde{s_N})$ where:

$$r_N(s_N) = \begin{cases} \rho_{s_N^2} s_N^1 & \text{if } \; s_N^2 \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{3.29}$$

We examine the three following cases.

i. $s_N^2 \geq \widetilde{s_N^2} \geq \theta$

$$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$

$$\rho_{s_N^2} s_N^1 \geq \rho_{\widetilde{s_N^2}} s_N^1 \Leftrightarrow$$

$$\frac{\beta(1 + s_N^2 - 2\theta)}{1 - \theta} s_N^1 \geq \frac{\beta(1 + \widetilde{s_N^2} - 2\theta)}{1 - \theta} s_N^1 \Leftrightarrow$$

$$s_N^2 \geq \widetilde{s_N^2}.$$

ii. $s_N^2 \geq \theta > \widetilde{s_N^2}$

$$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$

$$\rho_{s_N^2} s_N^1 \geq 0 \Leftrightarrow$$

$$\frac{\beta(1 + s_N^2 - 2\theta)}{1 - \theta} s_N^1 \geq 0 \Leftrightarrow$$

$$\frac{\beta(1 + s_N^2 - 2\theta)}{1 - \theta} s_N^1 \geq 0 \Leftrightarrow$$

$$1 + s_N^2 - 2\theta \geq 0.$$

Because $1 + s_N^2 - 2\theta \geq 1 + \theta - 2\theta = 1 - \theta$ and $1 - \theta \geq 0$ we can conclude that:

$$1 + s_N^2 - 2\theta \geq 0.$$

iii. $\theta > s_N^2 \geq \widetilde{s_N^2}$

$$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$
$$0 \geq 0.$$

We show that we **can not** prove the subadditivity condition and in turn, we proved that the stochastic SAIRPs violate the sufficient conditions for the optimality of a monotone policy in the second dimension of the state.

□

Theorem 1. *There exist optimal decision rules $d_t^* : S \to A_{s_t}$ for the stochastic SAIRP which are monotone non-increasing in the second dimension of the state for $t = 1, \dots, N-1$ if there is an upper-bound U on the number of batteries replaced at each decision epoch where $U = \frac{M\varepsilon}{2(1-s_t^2)}$, when M is the number of batteries at the swap station and ε is the discretized increment in capacity.*

Proof. We prove this monotonicity claim by showing the aforementioned five conditions are satisfied (Puterman, 2005). When we fix the first dimension of the state and action, in Lemma 1, we show that conditions i, ii, iii, and v are true. Thus, it suffices to prove condition iv is true in order to show the existence of monotone optimal decision rules for the second dimension. First, we prove the following claim and use it to show the subadditivity condition and in turn, monotonicity.

Claim 1. *j_A^2 and j_C^2 represents the same point if $a_t^2 - \widetilde{a_t^2} \leq U$.*

Proof. We know j_A^2 and j_C^2 represents the same point if $j_A^2 - j_C^2 \leq \frac{\varepsilon}{2}$ due to the precision in rounding. Thus,

$$j_A^2 - j_C^2 \leq \frac{\varepsilon}{2} \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] - \frac{1}{M}\left[s_t^2(M - \widetilde{a_t^2}) + \widetilde{a_t^2} - \delta^C(a_t^{1+} + a_t^{1-})\right] \leq \frac{\varepsilon}{2} \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(M - a_t^2 - (M - \widetilde{a_t^2}) + a_t^2 - \widetilde{a_t^2}\right] \leq \frac{\varepsilon}{2} \Leftrightarrow$$
$$\frac{1}{M}\left[s_t^2(-a_t^2 + \widetilde{a_t^2}) + a_t^2 - \widetilde{a_t^2}\right] \leq \frac{\varepsilon}{2} \Leftrightarrow$$
$$\frac{1}{M}\left[(a_t^2 - \widetilde{a_t^2})(1 - s_t^2)\right] \leq \frac{\varepsilon}{2} \Leftrightarrow$$
$$(a_t^2 - \widetilde{a_t^2}) \leq \frac{M\varepsilon}{2(1 - s_t^2)}.$$

Let $\widetilde{a_t^2} = 0$ to get the least upper-bound for the number of batteries to be replaced, we will have

$$a_t^2 \leq \frac{M\varepsilon}{2(1 - s_t^2)}.$$

□

With the case that $j_A^2 = j_C^2$, we have

$$j_A^2 = j_C^2 \geq j_B^2 \geq j_D^2.$$

Now, we revisit the condition, we seek to show that $q_t(k \mid s_t^2, a_t^2)$ is subadditive on $S^2 \times A'$ for every $k \in S^2$.

It suffices to show that

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}),$$

for every $k \in S^2$ if $s_t^2 \geq \widetilde{s_t^2}$ and $a_t^2 \geq \widetilde{a_t^2}$.

We divide S^2 space into four intervals and show the condition is satisfied for every interval.

i. $0 \leq k \leq j_D^2$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$
$$1 + 1 \leq 1 + 1.$$

ii. $j_D^2 < k \leq j_B^2$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$
$$1 + 0 \leq 1 + 1.$$

iii. $j_B^2 < k \leq j_C^2 = j_A^2$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$
$$1 + 0 \leq 1 + 0.$$

iv. $j_A^2 < k$

$$q_t(k \mid s_t^2, a_t^2) + q_t(k \mid \widetilde{s_t^2}, \widetilde{a_t^2}) \leq q_t(k \mid \widetilde{s_t^2}, a_t^2) + q_t(k \mid s_t^2, \widetilde{a_t^2}) \Leftrightarrow$$
$$0 + 0 \leq 0 + 0.$$

We conclude that $q_t(k \mid s_t^2, a_t^2)$ is subadditive on $S^2 \times A'$ for every $k \in S^2$. As all conditions are valid, we deduce that there exists monotone optimal decision rules in the second dimension of the state for the stochastic SAIRP when there is an upper-bound U on the number of batteries replaced at each decision epoch.

□

3.B Monotonicity of Value Functions

In this Appendix, we prove that the MDP value function for the stochastic SAIRP is monotone non-decreasing in the first, second, and both dimensions of the state. In Theorems 2 and 3, we show the monotonicity of the value function regarding s_t^2 and s_t^1, respectively, and in Theorem 4 we prove that the value function is monotone in (s_t^1, s_t^2). To prove these theorems, we need to ensure that four conditions are satisfied as given by Papadakia and Powell (2007) and Jiang and Powell (2015). These two articles use different notation, thus, for clarity, we define S as the state space, A as the action space, and the transition function as $f : S \times A \to S$. To prove the theorems, we first state key definitions used.

Definition 2. *Partial ordering operator $\preceq$ on the N-dimensional set S is defined as $s \preceq s'$ for any $s, s' \in S$, if $s(i) \leq s'(i)$ for all $i \in \{1, 2, \ldots N\}$ (Papadakia and Powell, 2007).*

Definition 3. *An N-dimensional real-valued function F is partially non-decreasing on the set S, if for all $s^-, s^+ \in S$ where $s^- \preceq s^+$, we have $F(s^-) \leq F(s^+)$ (Papadakia and Powell, 2007).*

Theorem 2. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in s_t^2.*

Proof. Fixing the first dimension of the state space, the MDP value function is monotone if the following four conditions are satisfied (Papadakia and Powell, 2007; Jiang and Powell, 2015).

1. For $e \succeq 0$ we have $f(s+e, a) \succeq f(s, a)$ for all $a \in A$ (Papadakia and Powell, 2007). Equivalently, if every $s_t^2, \widetilde{s_t^2} \in S^2$ with $s_t^2 \geq \widetilde{s_t^2}$, action $a_t = (a_t^1, a_t^2) \in A$ is taken, and demand equals D_t, the second state transition function f^2 satisfies:
$$f^2(s_t^1, s_t^2, a_t^1, a_t^2) \geq f^2(s_t^1, \widetilde{s_t^2}, a_t^1, a_t^2).$$
From Equation (3.24), if the beginning state is s_t^2, then f^2 equals j^2, and if the beginning state is $\widetilde{s_t^2}$, then f^2 equals $\widetilde{j^2}$. Because a_t^2, a_t^1, s_t^1 are constant, using Lemma 2 we prove our claim as
$$f^2(s_t^1, s_t^2, a_t^1, a_t^2) = j^2 \geq (s_t^1, \widetilde{s_t^2}, a_t^1, a_t^2) = \widetilde{j^2}.$$

2. The one period cost function $r_t(s, a)$ is partially non-decreasing in $s \in S$ for all $a \in A$, $t = 0, 1, \ldots, N-1$ (Papadakia and Powell, 2007). It is suffices to show for every $t < N$, $s_t^2, \widetilde{s_t^2} \in S^2$ with $s_t^2 \geq \widetilde{s_t^2}$, if we take action $a_t = (a_t^1, a_t^2) \in A$, then the one period reward function satisfies
$$r_t(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq r_t(s_t^1, \widetilde{s_t^2}, a_t^1, a_t^2, D_t). \tag{3.30}$$
We show that we can reduce Equation (3.30) to an always true statement.
$$\begin{gathered} r_t(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq r_t(s_t^1, \widetilde{s_t^2}, a_t^1, a_t^2, D_t) \Leftrightarrow \\ \tfrac{\beta(1+s_t^2-2\theta)}{1-\theta}\left[\min\{D_t, s_t^1 - a_t^{1-}\}\right] - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \geq \\ \tfrac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta}\left[\min\{D_t, s_t^1 - a_t^{1-}\}\right] - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \Leftrightarrow \end{gathered}$$

$$\frac{\beta(1+s_t^2-2\theta)}{1-\theta} \geq \frac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta} \Leftrightarrow$$
$$s_t^2 \geq \widetilde{s_t^2}.$$

3. The terminal cost function $r_N(s)$ is partially non-decreasing in $s \in S$ (Papadakia and Powell, 2007). It suffices to show that for every $s_N^2, \widetilde{s_N^2} \in S^2$ with $s_N^2 \geq \widetilde{s_N^2}$, that $r_N(s_N) \geq r_N(\widetilde{s_N})$. We proved this claim in condition (5) of Lemma 1 and Theorem 1.

4. For each $t < N$, s_t^2 and D_{t+1} are independent (Jiang and Powell, 2015).

   In our model, demand is a random variable and does not depend on the current state or action, including s_t^2.

As all the conditions are valid for the stochastic SAIRP, we can conclude that the value function is monotone in s_t^2.

□

Theorem 3. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in* s_t^1.

Proof. Fixing the second dimension of the state space, the MDP value function is monotone if the following four conditions are satisfied (Papadakia and Powell, 2007; Jiang and Powell, 2015).

1. For $e \succeq 0$ we have $f(s+e,a) \succeq f(s,a)$ for all $a \in A$ (Papadakia and Powell, 2007). It suffices to show for every $s_t^1, \widetilde{s_t^1} \in S^1$ with $s_t^1 \geq \widetilde{s_t^1}$, if we take action $a_t = (a_t^1, a_t^2) \in A$ and demand equals D_t, the first state transition function f^1 satisfies

$$f^1(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq f^1(\widetilde{s_t^1}, s_t^2, a_t^1, a_t^2, D_t).$$

   Using Equation (3.3), if the beginning state is s_t^1, then f^1 equals j^1, and if the starting state is $\widetilde{s_t^1}$, then f^1 equals $\widetilde{j^1}$. It suffices to show that

$$j^1 \geq \widetilde{j^1} \Leftrightarrow$$

$$s_t^1 + a_t^2 + a_t^{1+} - a_t^{1-} - \min\{D_t, s_t^1 - a_t^{1-}\} \geq \widetilde{s_t^1} + a_t^2 + a_t^{1+} - a_t^{1-} - \min\{D_t, \widetilde{s_t^1} - a_t^{1-}\} \Leftrightarrow$$
$$s_t^1 - \min\{D_t, s_t^1 - a_t^{1-}\} \geq \widetilde{s_t^1} - \min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}. \tag{3.31}$$

   Three cases can happen for the stochastic demand D_t, and we show that we can reduce Equation (3.31) to an always true statement in all cases.

   i. $s_t^1 - a_t^{1-} \geq \widetilde{s_t^1} - a_t^{1-} > D_t$

$$s_t^1 - D_t \geq \widetilde{s_t^1} - D_t \Leftrightarrow$$
$$s_t^1 \geq \widetilde{s_t^1}.$$

   ii. $s_t^1 - a_t^{1-} \geq D_t \geq \widetilde{s_t^1} - a_t^{1-}$

$$s_t^1 - D_t \geq \widetilde{s_t^1} - (\widetilde{s_t^1} - a_t^{1-}) \Leftrightarrow$$

$$s_t^1 - a_t^{1-} \geq D_t.$$

iii. $D_t > s_t^1 - a_t^{1-} \geq \widetilde{s_t^1} - a_t^{1-}$

$$s_t^1 - (s_t^1 - a_t^{1-}) \geq \widetilde{s_t^1} - (\widetilde{s_t^1} - a_t^{1-}) \Leftrightarrow$$

$$a_t^{1-} \geq a_t^{1-}.$$

From i, ii, and iii, we conclude that $j^1 \geq \widetilde{j}^1$. Thus,

$$f^1(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq f^1(\widetilde{s_t^1}, s_t^2, a_t^1, a_t^2, D_t).$$

2. The one period cost function $r_t(s,a)$ is partially non-decreasing in $s \in S$ for all $a \in A$, $t = 0, 1, \ldots, N-1$ (Papadakia and Powell, 2007). It suffices to show for every $t < N$, $s_t^1, \widetilde{s_t^1} \in S^1$ with $s_t^1 \geq \widetilde{s_t^1}$, if we take action $a_t = (a_t^1, a_t^2) \in A$, then the one period reward function satisfies
$$r_t(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq r_t(\widetilde{s_t^1}, s_t^2, a_t^1, a_t^2, D_t).$$
Using Equation (3.7), we reduce the following statement to an always true statement.
$$\begin{gathered} r_t(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq r_t(\widetilde{s_t^1}, s_t^2, a_t^1, a_t^2, D_t) \Leftrightarrow \\ \rho_{s_t^2}(\min\{D_t, s_t^1 - a_t^{1-}\}) - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \geq \\ \rho_{s_t^2}(\min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}) - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \Leftrightarrow \end{gathered}$$
$$\min\{D_t, s_t^1 - a_t^{1-}\} \geq \min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}. \tag{3.32}$$
The second dimension is fixed, so we could cancel out $\rho_{s_t^2}$ from both sides.
Similar to part 1, three cases can happen.

   i. $s_t^1 - a_t^{1-} \geq \widetilde{s_t^1} - a_t^{1-} > D_t$
   We can reduce Equation (3.32) to $D_t \geq D_t$ which is always true.

   ii. $s_t^1 - a_t^{1-} \geq D_t \geq \widetilde{s_t^1} - a_t^{1-}$
   We can reduce Equation (3.32) to $D_t \geq \widetilde{s_t^1} - a_t^{1-}$ which is true for this case.

   iii. $D_t > s_t^1 - a_t^{1-} \geq \widetilde{s_t^1} - a_t^{1-}$
   We can reduce Equation (3.32) to $s_t^1 \geq \widetilde{s_t^1}$ which is always true.

   So, we can conclude that
$$r_t(s_t^1, s_t^2, a_t^1, a_t^2, D_t) \geq r_t(\widetilde{s_t^1}, s_t^2, a_t^1, a_t^2, D_t).$$

3. The terminal cost function $r_N(s)$ is partially non-decreasing in $s \in S$ (Papadakia and Powell, 2007). It suffices to show that for every $s_N^1, \widetilde{s_N^1} \in S^1$ with $s_N^1 \geq \widetilde{s_N^1}$, that $r_N(s_N) \geq r_N(\widetilde{s_N})$.
As the second state is fixed equal to s_N^2, two possible cases can happen.

i. $s_N^2 < \theta$

$$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$
$$0 \geq 0.$$

ii. $s_N^2 \geq \theta$.

$$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$
$$\rho_{s_N^2} s_N^1 \geq \rho_{s_N^2} \widetilde{s_N^1}.$$

$\rho_{s_N^2}$ is not a function of the first dimension, so we can cancel it out from both sides.

$$s_N^1 \geq \widetilde{s_N^1}.$$

Thus, we conclude that

$$r_N(s_N) \geq r_N(\widetilde{s_N}).$$

4. For each $t < N$, s_t^1 and D_{t+1} are independent (Jiang and Powell, 2015). In our model, demand is a random variable and does not depend on the current state or action, including s_t^1.

As all the conditions are valid for the stochastic SAIRP, we conclude that the value function is monotone in s_t^1.

□

If there exists a monotone optimal policy, then Theorem 3 is implied. Widrick et al. (2018) proved that there exists a monotone optimal policy in s_t^1 *only* when demand is governed by a non-increasing discrete distribution. Thus, Theorem 3 provides a stronger result, as it does not depend on the probability distribution of demand.

Theorem 4. *The MDP value function of the stochastic SAIRP is monotonically non-decreasing in* (s_t^1, s_t^2).

Proof. The MDP value function is monotone if the following four conditions are satisfied (Papadakia and Powell, 2007; Jiang and Powell, 2015).

1. For $e \succeq 0$ we have $f(s+e,a) \succeq f(s,a)$ for all $a \in A$ (Papadakia and Powell, 2007). Using Definition 2 for partially ordered functions, it suffices to show that for every $s_t, \widetilde{s_t} \in (S^1 \times S^2)$ with $s_t^1 \geq \widetilde{s_t^1}$ and $s_t^2 \geq \widetilde{s_t^2}$, if we take action $a_t = (a_t^1, a_t^2) \in A$ and demand equals D_t, then the i^{th} state transition function f^i satisfies

$$f^i(s_t, a_t, D_t) \geq f^i(\widetilde{s_t}, a_t, D_t) \quad \forall i = 1,2.$$

We show the relationship between transition functions for each dimension.

i. $f^1(s_t, a_t, D_t) \geq f^1(\widetilde{s_t}, a_t, D_t)$
Using Equation (3.3), we can state that if the beginning state is s_t^1, then f^1 equals j^1 and if the starting state is $\widetilde{s_t^1}$, then f^1 equals $\widetilde{j^1}$. We proved the claim when $s_t^2 \geq \widetilde{s_t^2} \geq \theta$, in part 1 of Theorem 3. If $\theta > s_t^2 \geq \widetilde{s_t^2}$, as we are in the absorbing state, the only

possible action is $a_t = (0,0)$ that leads to $j^1 = s_t^1 \geq \widetilde{j}^1 = \widetilde{s_t^1}$. Similarly, if $s_t^2 \geq \theta > \widetilde{s_t^2}$, the only allowed action is $a_t = (0,0)$ because it is the only feasible action for both s_t^2 and $\widetilde{s_t^2}$. Thus, $j^1 = s_t^1 \geq \widetilde{j}^1 = \widetilde{s_t^1}$.

ii. $f^2(s_t, a_t, D_t) \geq f^2(\widetilde{s_t}, a_t, D_t)$
Using Equation (3.24), we reduce $f^2(s_t, a_t, D_t) \geq f^2(\widetilde{s_t}, a_t, D_t)$ to an always true statement:

$$f^2(s_t, a_t, D_t) \geq f^2(\widetilde{s_t}, a_t, D_t) \Leftrightarrow$$

$$\frac{1}{M}\left[s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \geq \frac{1}{M}\left[\widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-})\right] \Leftrightarrow$$

$$s_t^2(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \geq \widetilde{s_t^2}(M - a_t^2) + a_t^2 - \delta^C(a_t^{1+} + a_t^{1-}) \Leftrightarrow$$

$$s_t^2(M - a_t^2) \geq \widetilde{s_t^2}(M - a_t^2).$$

Because $M \geq \widetilde{a_t^2}$, we know

$$s_t^2 \geq \widetilde{s_t^2}.$$

The last statement is always true and we prove that $f^2(s_t, a_t, D_t) \geq f^2(\widetilde{s_t}, a_t, D_t)$.

2. The one period cost function $r_t(s, a)$ is partially non-decreasing in $s \in S$ for all $a \in A$, $t = 0, 1, \ldots, N-1$ (Papadakia and Powell, 2007). Using Definition 2 for partially ordered functions, it suffices to show that for every $s_t, \widetilde{s_t} \in (S^1 \times S^2)$ with $s_t^1 \geq \widetilde{s_t^1}$ and $s_t^2 \geq \widetilde{s_t^2}$, if we take action $a_t = (a_t^1, a_t^2) \in A$, then the one period reward function satisfies

$$r_t(s_t, a_t, D_t) \geq r_t(\widetilde{s_t}, a_t, D_t).$$

Using Equation (3.7), we reduce the following statement to an always true statement.

$$r_t(s_t, a_t, D_t) \geq r_t(\widetilde{s_t}, a_t, D_t) \Leftrightarrow$$

$$\rho_{s_t^2}(\min\{D_t, s_t^1 - a_t^{1-}\}) - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \geq$$

$$\rho_{\widetilde{s_t^2}}(\min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}) - K_t a_t^{1+} + J_t a_t^{1-} - L_t a_t^2 \Leftrightarrow$$

$$\left(\frac{\beta(1 + s_t^2 - 2\theta)}{1 - \theta}\right)(\min\{D_t, s_t^1 - a_t^{1-}\}) \geq \left(\frac{\beta(1 + \widetilde{s_t^2} - 2\theta)}{1 - \theta}\right)(\min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}) \Leftrightarrow$$

$$(1 + s_t^2 - 2\theta)(\min\{D_t, s_t^1 - a_t^{1-}\}) \geq (1 + \widetilde{s_t^2} - 2\theta)(\min\{D_t, \widetilde{s_t^1} - a_t^{1-}\}). \tag{3.33}$$

Three cases can happen for the stochastic demand D_t. We show that we can reduce Equation (3.33) to an always true statement in all cases.

i. $D_t > s_t^1 - a_t^{1-} \geq \widetilde{s_t^1} - a_t^{1-}$
Because $(1 + s_t^2 - 2\theta) \geq (1 + \widetilde{s_t^2} - 2\theta) \geq 0$ and $\min\{D_t, s_t^1 - a_t^{1-}\} \geq \min\{D_t, \widetilde{s_t^1} - a_t^{1-}\} \geq 0$, we can conclude that Equation (3.33) is always true.

ii. $s_t^1 - a_t^{1-} \geq D_t \geq \widetilde{s_t^1} - a_t^{1-}$

We know $(1+s_t^2-2\theta) \geq (1+\widetilde{s_t^2}-2\theta)$, thus it suffices to show that

$$(1+s_t^2-2\theta)(\min\{D_t, s_t^1-a_t^{1-}\}) \geq (1+s_t^2-2\theta)(\min\{D_t, \widetilde{s_t^1}-a_t^{1-}\}) \Leftrightarrow$$

$$\min\{D_t, s_t^1-a_t^{1-}\} \geq \min\{D_t, \widetilde{s_t^1}-a_t^{1-}\} \Leftrightarrow$$

$$D_t \geq \widetilde{s_t^1}-a_t^{1-}.$$

The last statement is always true for this case.

iii. $s_t^1-a_t^{1-} \geq \widetilde{s_t^1}-a_t^{1-} > D_t$
Using the same approach as the previous part, we can reduce Equation (3.33) to $D_t \geq D_t$ which is always true.

$$(1+s_t^2-2\theta)(\min\{D_t, s_t^1-a_t^{1-}\}) \geq (1+s_t^2-2\theta)(\min\{D_t, \widetilde{s_t^1}-a_t^{1-}\}) \Leftrightarrow$$

$$\min\{D_t, s_t^1-a_t^{1-}\} \geq \min\{D_t, \widetilde{s_t^1}-a_t^{1-}\} \Leftrightarrow$$

$$D_t \geq D_t.$$

So, we conclude that

$$r_t(s_t, a_t, D_t) \geq r_t(\widetilde{s_t}, a_t, D_t).$$

3. The terminal cost function $r_N(s)$ is partially non-decreasing in $s \in S$ (Papadakia and Powell, 2007). It suffices to show that for every $s_N, \widetilde{s_N} \in (S^1 \times S^2)$ with $s_N^1 \geq \widetilde{s_N^1}$ and $s_N^2 \geq \widetilde{s_N^2}$, that $r_N(s_N) \geq r_N(\widetilde{s_N})$. Using Equation (3.29), three cases can happen.

   i. $s_N^2 \geq \widetilde{s_N^2} \geq \theta$.

   $$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$

   $$\rho_{s_N^2} s_N^1 \geq \rho_{\widetilde{s_N^2}} \widetilde{s_N^1}.$$

   Because $\frac{\beta(1+s_t^2-2\theta)}{1-\theta} \geq \frac{\beta(1+\widetilde{s_t^2}-2\theta)}{1-\theta}$ and $s_N^1 \geq \widetilde{s_N^1}$ the last statement is true.

   ii. $s_N^2 \geq \theta > \widetilde{s_N^2}$

   $$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$

   $$\rho_{s_N^2} s_N^1 \geq 0.$$

   iii. $\theta > s_N^2 \geq \widetilde{s_N^2}$

   $$r_N(s_N) \geq r_N(\widetilde{s_N}) \Leftrightarrow$$

   $$0 \geq 0.$$

4. For each $t < N$, (s_t^1, s_t^2) and D_{t+1} are independent (Jiang and Powell, 2015).
In our model, demand is a random variable and does not depend on the current state or action, including (s_t^1, s_t^2).

As all the conditions are valid for the stochastic SAIRP, we conclude that the value function is monotone in (s_t^1, s_t^2). □

3.C Algorithmic and Experimental Parameter Settings

Parameter	Value	Description
$\overline{M}$	2	The starting number of batteries used for the small SAIRPs solved using BI
M	7	The number of batteries in the modest-sized SAIRPs
M	100	The number of batteries in the realistic-sized SAIRPs
$\overline{T}$	168	The time horizon (number of hours) in the small SAIRP
T	168	The time horizon (number of hours) in the modest-sized SAIRPs
T	744	The time horizon (number of hours) in the realistic-sized SAIRPs
θ	80%	The replacement threshold
ε	0.001	The capacity increment used in discretizing the 2^{nd} dimension of the state
$maxIteration+1$	3	The maximum number of regression-based initialization iterations
τ	500000	The maximum number of core ADP iterations in the modest-sized SAIRPs
τ	100000	The maximum number of core ADP iterations in the realistic-sized SAIRPs
w	25000	The harmonic stepsize parameter
μ_1	600	The STC stepsize parameter
μ_2	1000	The STC stepsize parameter
ζ	0.7	The STC stepsize parameter
α_0	1	The STC stepsize parameter

4. **Drones for Medical Delivery Considering Different Demands Classes: A Markov Decision Process Approach for Managing Health Centers Dispatching Medical Products**

Amin Asadi Sarah Nurre Pinkley

4.1 Introduction

In the last decade, there has been substantial growth in the use of drones for several applications, including but not limited to transportation, agriculture, and delivery (Mutzabaugh, 2017; Jensen, 2019; Weise, 2017; DHL Press Release, 2016). Specifically, delivery using drones has received extensive attention as it can reduce air pollution and traffic in congested areas (Dhote and Limbourg, 2020). Moreover, drones are a viable option to reach remote locations with inadequate road infrastructure (Davitt, 2019). During pandemics, drones provide a safe and low contact delivery method, which can effectively slow down the spread of the diseases (UNICEF Supply Division, 2020; McNabb, 2020). Many companies and organization, including Vanuatu's Ministry of Health and Civil Aviation (Kent, 2019), Zipline (Lyons, 2020), Matternet (Matternet, 2020), Manna Aero (Chandler, 2020) use drones to deliver and distribute medical supplies such as vaccines, medicine, and blood units. Effective operations of such companies, with a fleet of drones, require addressing battery-oriented issues, including limited flight range, long charging time needed, high price, and batteries' short life. In this research, we study a problem that considers the charge inside drone batteries and classify the stochastic demand according to drone flight range. 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 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 and Wishart, 2015). The faster batteries degrade the quicker the need for battery replacement which incurs a high cost and environmental waste. The application of a battery swap station is not limited to drones and can be extended to electric vehicles (Lambert, 2018; Fusheng, 2019), 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 (Dongmei, 2016; Chauvet, 2015; Peermohamed, 2017; Gordon-Bloomfield, 2014). 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. We classify the demand based on the distance between the hub and demand locations. Demand classification is widely used in the operations research community to properly capture characteristics of systems used in the supply chain, inventory control, and production management problems (Gayon et al., 2009; Benjaafar et al., 2011; Thompson et al., 2009; Mlinar and Chevalier, 2016). We consider a level of charge inside batteries corresponding to each 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 inside batteries. We formulate this problem as a stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD).

We use the Markov decision process (MDP) to model the stochastic SA-MCD. It is an appropriate modeling approach for problems like ours that are in the class of sequential decision-making under uncertainty problems (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 (RL) method to solve SA-MCDs. BI can provide exact solutions for problems like SA-MCDs 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 with an exploration feature (Powell, 2011) that is able to find high-quality approximate solutions for large-scale SA-MCDs, which are not solvable using BI. We show the convergence of our RL method and demonstrate its capacity to save computational time and memory.

We computationally test the SA-MCD model and solution methods on a case study influenced by 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 and convert them to the demand for drone missions, given that each drone can carry 2kg of medical products at a time (Baker, 2017). We classify this demand into two classes based on the distance between the station and each hospitals (i.e., closer hospitals are classified in level 1 demand and farther hospitals are classified in level 2 demand). We 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 SA-MCDs 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 RL method for a modest problem (15-21 drones). Solving the modest size problem shows 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.

Main Contributions. We summarize the main contribution of this chapter as follows. To the best of our knowledge:

- We are the first to propose stochastic scheduling and allocation problems with multiple classes of demand (SA-MCD) for managing operations of a drone swap station located at drone a hub. We classify the demand based on the distance between the station and hospitals generating the stochastic demands.
- We develop an MDP model for SA-MCD and solve it using backward induction and a reinforcement learning method to find optimal and near-optimal policies for the station that faces stochastic demand for sending drones to deliver medical supplies.
- We use the case study influenced by 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 our classification approach in modeling improves demand satisfaction, which is the primary purpose of delivering medical supplies using drones.

The remainder of this chapter is organized as follows. In Section 4.2, we discuss relevant literature to the modeling, application, and solution methods. In Section 4.3, we present our Markov Decision Process to model the stochastic scheduling and allocation problems with multiple classes of demand. In Section 4.4, we discuss the exact and approximate solution methods. In

Section 4.5, we outline the computational experiments conducted and provide insights for managing swap station operations. We summarize concluding remarks and propose directions for future work in Section 4.6.

4.2 Related Work

There is a growing interest in the use of drones for many different applications. We provide an overview of scientific works, which are more relevant to the model, application, and solution methods presented in this research. 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) solution methods.

Many researchers have studied managing swap station operations. In Chapter 2, we 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 the stochastic demand and battery degradation. We solve this problem using a heuristic, RL methods (Chapter 2), and a monotone approximate dynamic programming algorithm (Chapter 3) 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. The discussed papers do not consider different demand classes and multiple states of charge of batteries. 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-MCDs for managing internal operations in a swap station facing stochastic demand of different classes.

In recent years, there has been a rapid growth in using drones for many innovative applications (Macrina et al., 2020). Several papers (Barmpounakis et al., 2016; Chang and Lee, 2018;

Otto et al., 2018; Khoufi et al., 2019) 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 viable options for transportation and delivery (Otto et al., 2018; Dhote and Limbourg, 2020). Several companies are using drones to deliver medical supplies in different parts of the world (Kent, 2019; Lyons, 2020; Matternet, 2020; Chandler, 2020). Notably, we focus on a case study influenced by Zipline, a drone delivery company which 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 this year (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 operation time (Tokekar et al., 2013; Wang et al., 2017) and maximal flying distance (Savuran and Karakaya, 2016; Guerriero et al., 2014). In this research, we use the maximal coverage of 80km radius (160km 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 is in the class of sequential decision-making under uncertainty, and we use a Markov Decision Process (MDP) model, 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 closer to our research include drone applications (Al-Sabban et al., 2013; Baek et al., 2013; Fu et al., 2015), dynamic inventory and allocation (Federgruen and Zipkin, 1984; Somarin et al., 2017), and optimal timing of decisions (Alagoz et al., 2004; Chhatwal et al., 2010; Zhang et al., 2012; Khojandi et al., 2014). In SA-MCD, we classify the demands based on the distance between the station and demand nodes (hospitals) and link each class to one or multiple levels of charge inside drone batteries capable of satisfying the related demands.

Researchers broadly use demand classification 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 a MDP modeling approach. Gayon et al. (2009) provide optimal production policies for a supplier facing multiple classes of demands that are different in the 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. Mlinar 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. An instance of 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 incorporate this information to provide a framework that estimates the charging demands for charging stations and determine the service capacity of the stations without optimizing the system. As they do not consider optimization, this work is significantly different from our problem which considers optimization under uncertainty.

When an MDP model is large and complex it suffers from the curses of dimensionality (Powell, 2011; Sutton and Barto, 2018), and thus, is traditionally solved using approximate solution methods. Researchers apply Reinforcement Learning (RL) (or approximate dynamic programming (ADP), the term more used in the operations research community (Powell, 2011)) to find approximate solutions which are not solvable using standard exact solution methods, (e.g., dynamic programming (Puterman, 2005)). Examples of various ADP/RL methods in dynamic allocation problems are temporal difference (Roy et al., 1997; Çimen and Kirkbride, 2017, 2013), case-based myopic RL (Jiang and Sheng, 2009), Q-Learning (Chaharsooghi et al., 2008), value function approximation (Bertsimas and Demir, 2002; Erdelyi and Topaloglu, 2010; Maxwell et al., 2010), linear function approximations (Powell and Topaloglu, 2005), and policy iteration (Nasrollahzadeh et al., 2018). In this research, we apply a value function approximation using a look-up table (e.g., (Jiang and Sheng, 2009; Kwon et al., 2008)) with an ε-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 (increase the exploitation rate) to make the algorithm converge as it proceeds toward iterations. In Section 4.4.2, we present a comprehensive explanation of our RL method.

4.3 Problem Description and Formulation

In this section, we present our Markov Decision Process (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 by 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 demand class. As follows, the uncertainty in our MDP is the number of requests (i.e., demand) for each demand class by 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 Figure 4.1.

We link each demand class with the required amount of battery charge that is necessary to

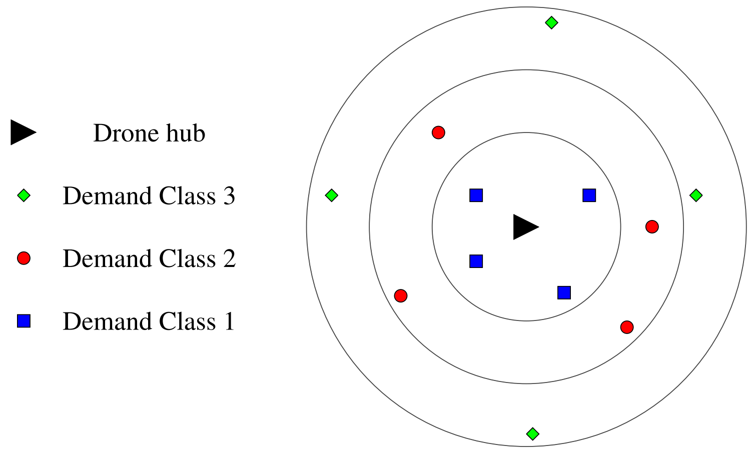


Figure 4.1: An example of demand classification based on the distance between the location of demand and the drone hub.

make the round-trip flight from the drone hub to the medical facility and back. 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. Thus, we make decisions about how many batteries are recharged to different charge levels over time. Our system incorporates time-varying elements, including the mean demand per class over time. As a result, we use a finite horizon MDP. We seek to maximize the expected total reward wherein the reward equals the summation over all demand classes and time periods of the weighted met demand. We note that the main purpose of our swap station is demand satisfaction; thus, we do not directly incorporate different charging costs and times into our model. However, in the objective function, we use multipliers for the amount of demand of each class that is satisfied using the batteries of higher charge levels. In this design, higher charge level batteries are capable of satisfying demand for lower classes, but this can be penalized as it caused unnecessary charging costs. To maximize the expected total weighted demand, we seek optimal policies indicating how many batteries are charged to each charge level by state and time. We proceed by detailing the specific components of our MDP.

Decision Epochs: Discrete times within the finite time horizon $N < \infty$ in which we make decisions. The set of decision epochs is $T = \{1, 2, \dots, N-1\}$.

States: The state of the system, s_t, is dynamic and defined in the C-dimensional state space S. Thus, $s_t = (s_t^1, s_t^2, \ldots, s_t^C) \in S = (S^1 \times S^2 \times \ldots S^C)$, where S^i is the state space for battery charge level i for $i = 1, \ldots, C$. Each $S^i = \{0, \ldots, M\}$ where M represents the total number of batteries. For each dimension, $s_t^i \in S^i$ equals the number of batteries with i level charge. The total number of batteries over all charge levels must not exceed M in accordance with Equation (4.1). All batteries with a charge level lower than the lowest charge level (i.e., level 1) are implicitly denoted as being level 0 which does not need to be stored but instead, can be calculated as $s_t^0 = M - \sum_{i=1}^{C} s_t^i$.

$$S = \left\{ (s_t^1, s_t^2, \ldots s_t^C) : \left(\sum_{i=1}^{C} s_t^i \leq M, \quad \forall t \in T \right) \right\}. \tag{4.1}$$

The C-dimensional state space corresponds to the C demand classes. As previously mentioned, each demand class represents a set of medical facilities with similar round-trip delivery flight times. We link the state space with these demand classes by stating that a drone powered with charge level i is able to satisfy requests from demand class i and lower. In other words, a request from demand class i is able to be satisfied by a drone powered with charge level i or higher. We make the assumption that a demand request from class i is 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 $2, \ldots, C$ 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 to assign a battery with charge level 3, and continue increasing the charge level until a battery is available. 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 $({}^{C(C-1)}\!/_2)$ such that a_t^{ik} represents the number of batteries starting at charge level i which are recharged to level k for $k > i$ at time t. As follows,

$$a_t = \{a_t^{ik} \in A_{s_t}^{ik} : \ \forall i = 0, 1, \ldots, C-1, \ k = i+1, \ldots, C\} \tag{4.2}$$

where

$$A^{ik}_{s_t} = \{0, 1, \ldots, s^i_t\} \qquad \forall i = 1, \ldots, C-1,\ k = i+1, \ldots, C, \text{ and} \tag{4.3}$$

$$A^{0k}_{s_t} = \{0, 1, \ldots, M - \sum_{i=1}^{C} s^i_t\} \quad \forall k = 1, \ldots, C. \tag{4.4}$$

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 Equations (4.5) and (4.6).

$$\sum_{k=1}^{C} a^{0k}_t \le M - \sum_{i=1}^{C} s^i_t. \tag{4.5}$$

$$\sum_{k=i+1}^{C} a^{ik}_t \le s^i_t \qquad \forall i = 1, \ldots, C-1,\ \forall t \in T. \tag{4.6}$$

In Figure 4.2, 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.

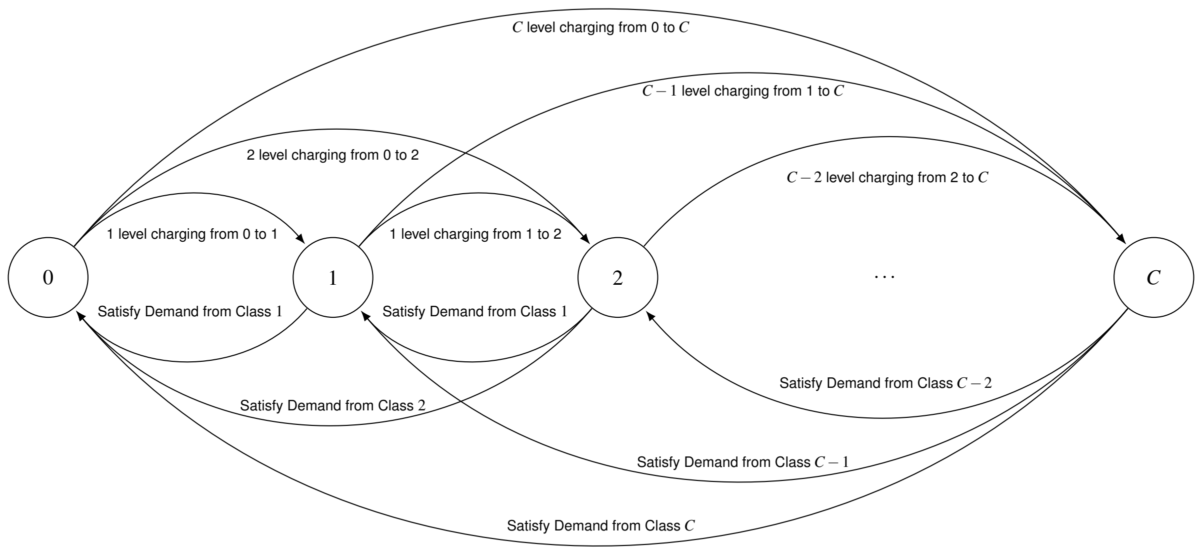

C level charging from 0 to C
$C-1$ level charging from 1 to C
2 level charging from 0 to 2
$C-2$ level charging from 2 to C
1 level charging from 0 to 1
1 level charging from 1 to 2
0
1
2
$\cdots$
C
Satisfy Demand from Class 1
Satisfy Demand from Class 1
Satisfy Demand from Class $C-2$
Satisfy Demand from Class 2
Satisfy Demand from Class $C-1$
Satisfy Demand from Class C

Figure 4.2: An instance of state transition for a single battery.

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. The state transitions and the probability transition function is complex; to illustrate these functions, we focus on an MDP where $C = 2$. 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 as $L = (L^1, L^2)$ and allow this to represent the number of batteries within the two classes after all actions are taken and batteries are used 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). 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 Equations (4.7) and (4.8). In Equation (4.7), $\min\{s_t^1 - a_t^{12}, D_t^1\}$ equals the satisfied demand of class 1 using the available batteries with charge level 1. Similarly, $\min\{s_t^2, D_t^2\}$ denotes the satisfied demands of class 2 using batteries with charge level 2 in Equation (4.8).

$$L^1 = s_t^1 + a_t^{01} - a_t^{12} - \min\{s_t^1 - a_t^{12}, D_t^1\}. \tag{4.7}$$

$$L^2 = s_t^2 + a_t^{02} + a_t^{12} - \min\{s_t^2, D_t^2\}. \tag{4.8}$$

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 Equations (4.9) and (4.10).

$$s_{t+1}^1 = L^1 + \min\big\{\max\{0, D_t^1 - (s_t^1 - a_t^{12})\}, \max\{0, s_t^2 - D_t^2)\}\big\}. \tag{4.9}$$

$$s_{t+1}^2 = L^2 - \min\big\{\max\{0, D_t^1 - (s_t^1 - a_t^{12})\}, \max\{0, s_t^2 - D_t^2)\}\big\}. \tag{4.10}$$

In Equations (4.9) and (4.10), $V^1 = \max\{0, D_t^1 - (s_t^1 - a_t^{12})\}$ is the amount of unsatisfied class

1 demand after using level 1 charged batteries and $V^2 = \max\{0, s_t^2 - D_t^2)\}$ is the number of left-over 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 (V^1, V^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 only introduce the intermediate state of the system to clarify the state transitions, transition probability, and immediate reward function.

Now, we present the transition probability function from state s_t to state $s_{t+1} = j = (j^1, j^2)$ using Equation (4.11).

$$p(j|s_t, a_t) = \begin{cases} (p^1_{s_t^1 + a_t^{01} - a_t^{12} - j^1})(p^2_{s_t^2 + a_t^{02} + a_t^{12} - j^2}) & \text{if } a_t^{01} < L^1 \le s_t^1 + a_t^{01} - a_t^{12}, \\ & a_t^{01} + a_t^{12} < L^2 \le s_t^2 + a_t^{02} + a_t^{12}, \\ & j^2 = L^2, \text{ and } j^1 = L^1 \\ (p^1_{s_t^1 + a_t^{01} - a_t^{12} - j^1})(q^2_{s_t^2}) & \text{if } a_t^{01} < L^1 \le s_t^1 + a_t^{01} - a_t^{12}, \\ & a_t^{01} + a_t^{12} = L^2, j^2 = L^2, \text{ and } j^1 = L^1 \\ (q^1_{s_t^1 - a_t^{12}})(q^2_{s_t^2}) & \text{if } a_t^{01} = L^1, a_t^{01} + a_t^{12} = L^2, \\ & j^2 = L^2, \text{ and } j^1 = L^1 \\ (p^1_{s_t^1 - a_t^{12} + L^2 - j^2})(p^2_{s_t^2 + a_t^{02} + a_t^{12} - L^2}) & \text{if } a_t^{01} = L^1, a_t^{01} + a_t^{12} < L^2 \le s_t^2 + a_t^{02} + a_t^{12}, \\ & a_t^{02} + a_t^{12} < j^2 \le L^2, \text{ and } j^1 = a_t^{01} + L_t^2 - j^2 \\ (q^1_{s_t^1 - a_t^{12} + L^2 - j^2})(p^2_{s_t^2 + a_t^{02} + a_t^{12} - L^2}) & \text{if } a_t^{01} = L^1, a_t^{01} + a_t^{12} < L^2 \le s_t^2 + a_t^{02} + a_t^{12}, \\ & a_t^{02} + a_t^{12} = j^2, \text{ and } j^1 = a_t^{01} + L_t^2 - j^2 \\ 0 & \text{otherwise,} \end{cases} \tag{4.11}$$

where $p_x^i = P(D_t^i = x)$ and $q_x^i = \sum_{u=x}^{\infty} p_u^i = P(D_t^i \ge x)$ $\forall i = 1, 2$. In all of the cases in Equation

(4.11), the intermediate state transitions are calculated using Equations (4.7) and (4.8). The first case in Equation (4.11) 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. The difference is 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 can not be satisfied in the first and second stage of the demand satisfaction process.

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 Equation (4.12)

$$r_t(s_t, a_t, j) = \rho^{11}(s_t^1 + a_t^{01} - a_t^{12} - L^1) + \rho^{21}(L^2 - j^2) + \rho^{22}(s_t^2 + a_t^{02} + a_t^{12} - L^2) \tag{4.12}$$

for $t = 1, \ldots, N-1$. The immediate reward is a function of weighted met demand. We use ρ^{ij} to put weights on the amount of met demand of class j using level i charged batteries. We can adjust the value of multipliers ρ^{ij} based on different factors (e.g., cost of charging) to change the penalty/reward of satisfying class j demands using level i charged batteries. For instance, when we implicitly consider the cost factor, we can set $\rho^{ij} > \rho^{i'j}$ where $i < i'$ as we incur a higher cost to recharge batteries to level i' than level i, therefore the reward generated to satisfy class j demand is lower using level i' charged than level i batteries. In other words, we penalize the reward function by using a lower value for ρ^{ij}s when higher-level charged batteries are used to satisfy class j demand.

In the first term of Equation (4.12), $(s_t^1 + a_t^{01} - a_t^{12} - L^1)$ denotes the number of level 1 charged drones used to satisfy class 1 demand. In the second term, $(L^2 - j^2)$ equals to the number of level 2 charged batteries used to satisfy class 1 demand. In the third term, $(s_t^2 + a_t^{02} + a_t^{12} - L^2)$ 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 ρ^{ij}, we implicitly include a cost factor via assigning a penalty/reward to demand satisfaction with an excessive level of charge. For instance, when $\rho^{21} = \rho^{22} = \rho^{11} = 1$, 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 $\rho^{11} = 2\rho^{21}$, which means we penalize the reward of satisfying class 1 demand with level 2 charged batteries.

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. With this, we assume that there is sufficient demand for each level. Thus, we define the terminal reward using Equation (4.13).

$$r_N(s_N) = \begin{cases} \rho_N^{11} s_N^1 + \rho_N^{22} s_N^2 & \text{if } s_N^1 + s_N^2 \leq M, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

We calculate the immediate expected reward $r_t(s_t, a_t)$, using the immediate reward and transition probability functions given by Equation (4.14),

$$r_t(s_t,a_t) = \sum_{j,L\in S}\Big[p_t(j|s_t,a_t)\big(\rho^{11}(s_t^1+a_t^{01}-a_t^{12}-L^1)+\rho^{21}(L^2-j^2)+ \rho^{22}(s_t^2+a_t^{02}+a_t^{12}-L^2)\big)\Big]. \tag{4.14}$$

We derive the decision rules, $d_t(s_t) : s_t \to A_{s_t}$, from the action set to maximize the total expected reward. 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 π is a sequence of decision rules for all decision epochs, that is $d_t^{\pi}(s_t)\ \ \forall\, t \in T$. We can calculate the expected total reward of policy π for the problems starting from an arbitrary initial state s_1 using Equation (4.15). The optimal policy, π^*, maximizes the expected total reward.

$$V_N^{\pi}(s_1) = \mathbb{E}_{s_1}^{\pi}\left[\sum_{t=1}^{N-1} r_t(s_t,a_t)+r_N(s_N)\right]. \tag{4.15}$$

4.4 Solution Methodology

4.4.1 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); thus, backward induction (BI) can determine such policy (number of recharging actions) that maximizes the expected total reward or weighted met demand over time. Let $V_t^*(s_t)$ be the optimal value function equivalent to the maximum expected total reward from decision epoch t onward when the system is in state s_t. Then, we can use optimality (Bellman) equations, given by Equation (4.16), 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 Equation (4.13). Then, the algorithm starts from the last decision epoch and finds the optimal actions and corresponding values using Equations (4.16) and (4.17) stepping backward in time. The algorithm aims to find the optimal expected total reward over the time horizon, $V_1^*(s_1)$, 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.

$$V_t(s_t) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p_t(j \mid s_t, a_t) u_{t+1}(j) \right\}. \tag{4.16}$$

$$a^*_{s_t,t} = \text{arg max}_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \sum_{j \in S} p_t(j|s_t, a_t) u_{t+1}(j) \right\}. \tag{4.17}$$

The size of state space, action space, transition probability, and optimal policies are functions of $O(M^2)$ and $O(M^3)$, $O(M^7N)$, $O(M^2N)$, respectively. As the size of the problem increases, it becomes challenging for BI to find the optimal solution due to the curses of dimensionality which cause computational time and memory issues. Hence, we proceed with presenting our reinforcement learning (RL) method, which is capable of circumventing such problems (Powell, 2011; Sutton and Barto, 2018).

4.4.2 Reinforcement Learning

In this section, we explain the Reinforcement Learning (RL) method used to find high-quality approximate solutions and overcome the curses of dimensionality (Powell, 2011) of the stochastic scheduling and allocation problem with multiple classes of demand (SA-MCD). We proceed by introducing the notation and continue with our RL features and procedure.

In our RL method, first, we determine the number of drones (M), decision epochs ($N - 1$), (τ_1) iterations, and (τ_2) sample paths. 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 Equation (4.13). For every iteration, we select an initial state s_1^n. To select the action, we use the ε-greedy method (Powell, 2011) that allows exploring the action space that works as follows. We generate a random number, *Rand*. Then, we compare *Rand* with the exploration rate, ε^n, at iteration n. If *Rand* $< \varepsilon^n$, we select a feasible action arbitrarily. Otherwise, we generate τ_2 sample paths of

Table 4.1: Notation used in the reinforcement learning algorithm.

Notation	Description
τ_1	The number of core RL iterations
τ_2	The number of sample paths of demands (realized uncertainty)
$V_t^n(s_t)$	The optimal value of being in state s_t at time t for iteration n
$\overline{V}_t^n(s_t)$	The approximate value of being in state s_t at time t for iteration n
$\hat{\upsilon}_t^n(s_t)$	The observed value of state s_t at time t for iteration n
α_n	The step-size value at iteration n
ε^n	The exploration rate at iteration n
Rand	A random number that is used to select exploration or exploitation
$z_t^n(s_t^n)$	The smoothed value of being in state s_t at time t for iteration n

demands (realized uncertainty) and select the action that maximizes the observed value $\hat{\upsilon}_t^n(s_t)$ according to Equations (4.18) and (4.19), wherein $\overline{V}_{t+1}^{n-1}(s_{t+1})$ is used to approximate the value of $\mathbb{E}(V_{t+1} \mid s_t, a_t)$ for each sample path. If an action, is selected over multiple sample paths, we use the average of $\overline{V}_{t+1}^{n-1}(s_{t+1})$ as the approximation. The observed value and the approximated value at the previous iteration are smoothed using a step size 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 new iteration starts. The same process is repeated until τ_1 iterations are completed.

$$a_{s_t,t}^n = \underset{a_t \in A_{s_t}}{\arg\max} \left\{ r_t(s_t, a_t) + \overline{V}_{t+1}^{n-1}(s_{t+1}) \right\}. \tag{4.18}$$

$$\hat{\upsilon}_t^n(s_t^n) = \max_{a_t \in A_{s_t}} \left\{ r_t(s_t, a_t) + \mathbb{E}(V_{t+1} \mid s_t, a_t) \right\}. \tag{4.19}$$

4.5 Computational Results

In this section, we explain the results of solving stochastic scheduling and allocation problems with multiple classes of demand (SA-MCD) using realistic data influenced by the drone deliv-

Algorithm 4 Reinforcement Learning Method

```
 1: Initialize $M$ drones, $N-1$ decision epochs, $\tau_1$ iterations, and $\tau_2$ sample paths
 2: Set $\overline{V}_N^n(s) = r_N(s)$ for $s \in S$ and $n = 1,\ldots,\tau_1$
 3: Set $n = 1$
 4: while $n \leq \tau_1$ do
 5:     Select initial state $s_1^n$
 6:     for $t = 1,\ldots,N-1$ do
 7:         Generate a random number $Rand$
 8:         if $Rand < \varepsilon^n$ then
 9:             Sample an observation of the uncertainty, $D_t$
10:             Determine a random feasible action, $a_n^t$
11:             Calculate the observed value, $\hat{\upsilon}_t^n(s_t^n)$
12:         else
13:             Find an action that maximizes $\hat{\upsilon}_t^n(s_t)$ over $\tau_2$ sample paths
14:             Sample an observation of the uncertainty, $D_t$
15:             Calculate the observed value, $\hat{\upsilon}_t^n(s_t^n)$
16:         end if
17:         Smooth the new observation with the previous approximated value,
                $$z_t^n(s_t^n) = (1-\alpha_n)\overline{V}_t^{n-1}(s_t^n) + \alpha_n\hat{\upsilon}_t^n(s_t^n) \qquad (4.20)$$
18:         Update the present approximation using the smoothed value, $\overline{V}_t^n(s_t^n) \leftarrow z_t^n(s_t^n)$
19:         Determine next state, $s_{t+1}^n$
20:     end for
21:     Increment $n = n+1$
22: end while
```

ery 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-MCDs (15-21 drones) using exact solution methods. 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 provides near-optimal solutions for the modest instances and can 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. We proceed by first explaining the data.

4.5.1 Data

In this research, we consider the medical supply delivery by Zipline, a drone delivery company operating in Rwanda. 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 4.2, 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 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 80km (Ackerman, 2020), we assume demands of hospitals located within [0km, 40km) and [40km, 80km] 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 4.2) from the 33 identified hospitals as they are located at a distance out of the drone's flight range from the origin, Zipline station, which drones with fully-charged batteries can not cover. We consider ten hospitals in class 1 and 17 hospitals in class 2 which are located throughout 15 distinct districts in Rwanda.

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 10km 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 Figure 4.3, we display the geographical locations of airports, hospitals, and the Zipline station.

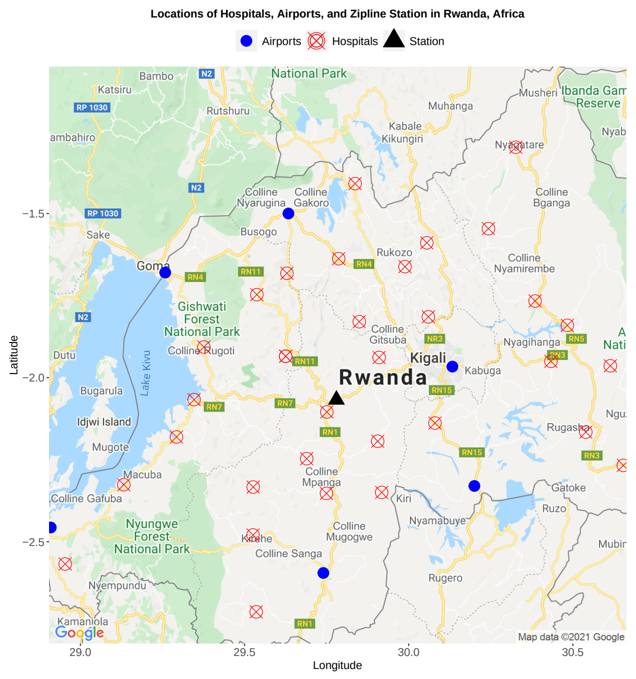
Consistent with (Swartzman, 1970; Armony et al., 2015), we use a non-homogenous Poisson process to determine the patients' arrival to hospitals. We examine the daily operations of the

Table 4.2: The data associated with blood unit delivery using Zipline drones in Rwanda, Africa.

Hospital name	District	Dist. to Zipline station (km)	Pop. reach the hospital	Pop. need blood unit/year	Pop. need blood unit /day	Rounded # of flights needed/day	Class of demand
Nyamata	Bugesera	34.3	361914	7238.3	19.8	10	1
Butaro	Burera	73.5	336582	6731.6	18.4	10	2
Nemba	Gakenke	47.8	169117	3382.3	9.3	5	2
Ruli	Gakenke	27.6	169117	3382.3	9.3	5	1
Kiziguro	Gatsibo	75.8	216510	4330.2	11.9	6	2
Ngarama	Gatsibo	77.5	216510	4330.2	11.9	6	2
Byumba	Gicumbi	61.3	395606	7912.1	21.7	11	2
Gakoma	Gisagara	34.9	161253	3225.1	8.8	5	1
Remera Rukoma	Kamonyi	20.3	340501	6810.0	18.7	10	1
Kibuye Referral	Karongi	48.2	110603	2212.1	6.1	4	2
Kirinda	Karongi	22.6	110603	2212.1	6.1	4	1
Mugonero	Karongi	55.6	110603	2212.1	6.1	4	2
Gahini	Kayonza	85.4	172079	3441.6	9.4	5	NA
Rwinkwavu	Kayonza	94.0	172079	3441.6	9.4	5	NA
Kirehe	Kirehe	99.6	340368	6807.4	18.7	10	NA
Kabgayi	Muhanga	5.2	319141	6382.8	17.5	9	1
Kibungo	Ngoma	85.1	336928	6738.6	18.5	10	NA
Muhororo	Ngororero	22.6	333713	6674.3	18.3	10	1
Shyira	Nyabihu	46.0	294740	5894.8	16.2	9	2
Nyagatare	Nyagatare	104.9	465855	9317.1	25.5	13	NA
Kaduha	Nyamagabe	40.8	170746	3414.9	9.4	5	2
Kigeme	Nyamagabe	53.8	170746	3414.9	9.4	5	2
Kibogora	Nyamasheke	77.5	190902	3818.0	10.5	6	2
Nyanza	Nyanza	31.9	323719	6474.4	17.7	9	1
Munini	Nyaruguru	76.8	294334	5886.7	16.1	9	2
Kabaya	Rubavu	44.7	201831	4036.6	11.1	6	2
Gitwe	Ruhango	22.4	159943	3198.9	8.8	5	1
Ruhango	Ruhango	19.8	159943	3198.9	8.8	5	1
Kinihira	Rulindo	50.7	143841	2876.8	7.9	4	2
Rutongo	Rulindo	41.9	143841	2876.8	7.9	4	2
Mibilizi	Rusizi	107.3	200429	4008.6	11.0	6	NA
Murunda	Rutsiro	48.3	324654	6493.1	17.8	9	2
Rwamagana	Rwamagana	73.7	313461	6269.2	17.2	9	2

drone station wherein the time between two consecutive decision epoch is 90 minutes (1.5hr). Thus, $N = {}^{24}/_{1.5} + 1 = 17$. 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 127km/hr (Petrova and 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



Locations of Hospitals, Airports, and Zipline Station in Rwanda, Africa
Airports
Hospitals
Station
Latitude
Longitude
Rwanda
Kigali
Lake Kivu

Figure 4.3: Locations of hospitals (demand nodes), the swap station located in Zipline drone hub, and airports in Rwanda.

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 yielded 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; Jones et al., 2007), to derive the mean demand for blood units of time t over a day. The pattern in the literature indicates 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 of time t. In Figure 4.4, 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 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. Finally, the mean demand for either demand class, λ_t^1, λ_t^2, is the aggregation of mean demands for flights from the hospitals within each demand class.

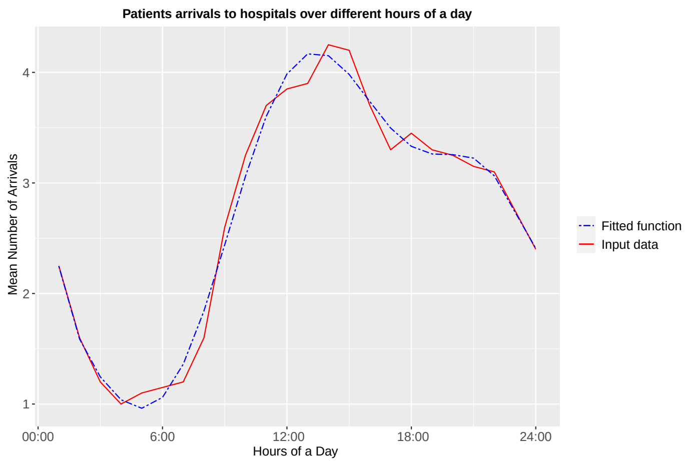


Figure 4.4: Pattern of patients arrivals to a hospital over a day.

In the first experiment, we consider Zipline has a fleet of 15 drones (Staedter, 2016). We set $\rho^{11} = \rho^{22} = 1$ and $\rho^{21} = 0.5$ that indicates 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. In other words, with this setting, the company receives more reward when batteries with level 2 and level 1 charge are used to satisfy class 2 and class 1 demand, respectively, instead of using level 2 charged batteries to satisfy both classes of demand.

The setting of our RL parameters is as follows. The number of core iterations is $\tau_1 = 200000$. 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 over our experiments. We test $\tau_2 = 1, 5, 10, \ldots, 50$ 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 τ_2 only magnifies the computational time with little improvement in the quality of the result; thus, we set $\tau_2 = 30$. We use the adaptive stepsize function provided by George and Powell (2006). We use $\varepsilon^n = {}^1\!/\!{}_n$ to adjust the value of the exploration rate at iteration n used in the ε-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.

4.5.2 Discussion and Analysis

In this section, first, we feed the data explained in Section 4.5.1 to solve the problem using exact and approximate solution methods, Backward Induction (BI) and Reinforcement Learning (RL), respectively. Then, we analyze the optimal policies (BI solutions) and assess the quality of near-optimal solutions derived from RL. Moreover, as the drone delivery company can control and adjust ρ^{21} (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 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 768GB of memory for running all of our computational tests.

4.5.2.1 Comparing Results of BI and RL

In this section, we present the results from solving stochastic scheduling and allocation problems with multiple classes of demand (SA-MCD) using BI and RL using the data presented in Section 4.5.1. 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 minutes. That is, the decisions are made at 16 decision epochs, t, where t = 00:00, 1:30, 3:00, ..., 10:30, 12:00, 13:30, ..., 22:30. We calculate the average percentage of met demand using Equations (4.21) and (4.22) wherein we generate 500 sample paths of realized demand using the Poisson distribution at time t for each class of demand.

$$(\%) \text{ of Demand Met over Time for a Sample Path} = \left| \frac{\text{Tot. \# Met Dem. over Time - Tot. \# Realized Dem. over Time}}{\text{Tot. \# Realized Dem. over Time}} \right| * 100\%. \quad (4.21)$$

$$\text{Average } (\%) \text{ of Demand Met} = \frac{\sum_{i=1}^{\text{\# of Sample Paths}} (\%) \text{ of Met Demand over Time for Sample Path } i}{\text{\# of Sample Paths}}. \quad (4.22)$$

We summarize the results in Table 4.3. 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 4.3. We provide more detailed results about demand satisfaction by class later in Table 4.5. As shown, RL is faster and can generate a high-quality solution with 5.3% of optimality gap (derived from Equation (4.23)) in 8 minutes. In Figure 4.5, we show the convergence of our proposed RL method.

$$\text{Optimality Gap} = \left| \frac{\text{Exp. Tot. Reward BI - Exp.Tot. Reward RL Method}}{\text{Exp. Tot. Reward BI}} \right| * 100\%. \quad (4.23)$$

Table 4.3: The expected total reward and computational time of solving SA-MCD with $\rho^{21} = 0.5$ and $M = 15$ using BI and RL.

Solution Method	Expected Total Reward	Average Met Demand (%)	Computational Time (s)
BI	115.1	63.7	6740.7
RL	109.0	60.9	483.7

In Table 4.4, we provide the summary of the results from solving the problem with 15 to 21 drones using BI and RL. We note, BI cannot find the optimal solution when the number of drones is greater than 21. For 15-21 drones, RL provides an optimality gap of less than 6% for all instances and significantly reduces computational time. The maximum difference between the average percentage of met demand of RL and BI is less than 5%. The results indicate the high performance of our RL method in providing approximate solutions.

Table 4.4: Computational time, memory used, and average percentage of met demand over time for 500 sample paths when $\rho^{21} = 0.5$ using BI and RL methods.

	Backward Induction (BI)			Reinforcement Learning (RL)			
M	Average Met Demand (%)	Comput. Time (s)	Memory Used (GB)	Average Met Demand (%)	Comput. Time (s)	Memory Used (GB)	Opt. Gap (%)
15	63.7	6740.7	34.2	60.9	483.7	13.4	5.3
16	66.9	11630.4	68.1	64.3	494.1	13.4	3.3
17	69.9	19356.4	125.8	65.2	517.0	13.4	5.0
18	72.6	31405.8	192.2	70.6	534.5	13.4	3.4
19	75.4	49812.5	287.5	72.6	575.7	13.4	3.5
20	77.9	77170.6	411.9	73.6	598.1	13.4	4.8
21	80.2	117154.0	621.1	78.3	600.1	13.4	2.7

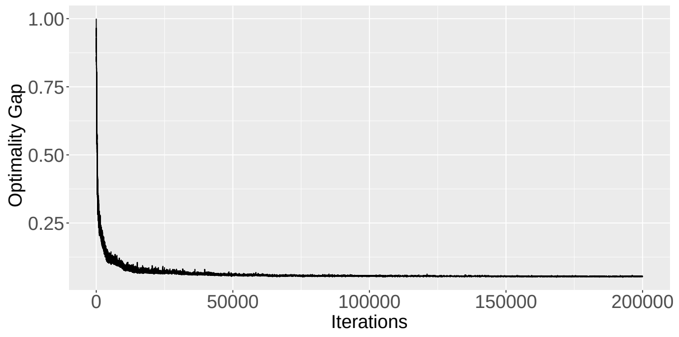


Figure 4.5: Expected total reward convergence of RL method.

We can find the sample paths of visited states and policies using the sample paths of realized demands. Given the initial state, taken actions, realized demand, and the state transition functions given by Equations (4.9) and (4.10), we can find the future visited state. The consecutive visited states form the sample path of states. The sample paths of policies are the consecutive selected actions derived from the BI and RL solution methods.

In Figure 4.6, the first, second, and third row depict sample paths of states, optimal policy, and met demands when the stochastic demand equals mean demand at time t and $\rho^{21} = 0.5, 1,$ and 2, respectively, for $M = 15$. Intuitively, when $\rho^{21} = 0.5$, the level 1 charged batteries are used to satisfy class 1 demand and $a_t^{12} = 0$. When ρ^{21} increases more batteries of class 1 are recharged to class 2 to be used for satisfying the demand of class 1. When $\rho^{21} = 1$, no batteries are recharged to level 1 because there is no difference between the value of satisfying the class 1 demand using level 1 and level 2 charged batteries. Hence, the optimal policy includes recharging batteries to level 2 that can be used for either classes of demand. For all values of ρ, we observe more recharging actions when demands are in peak periods.

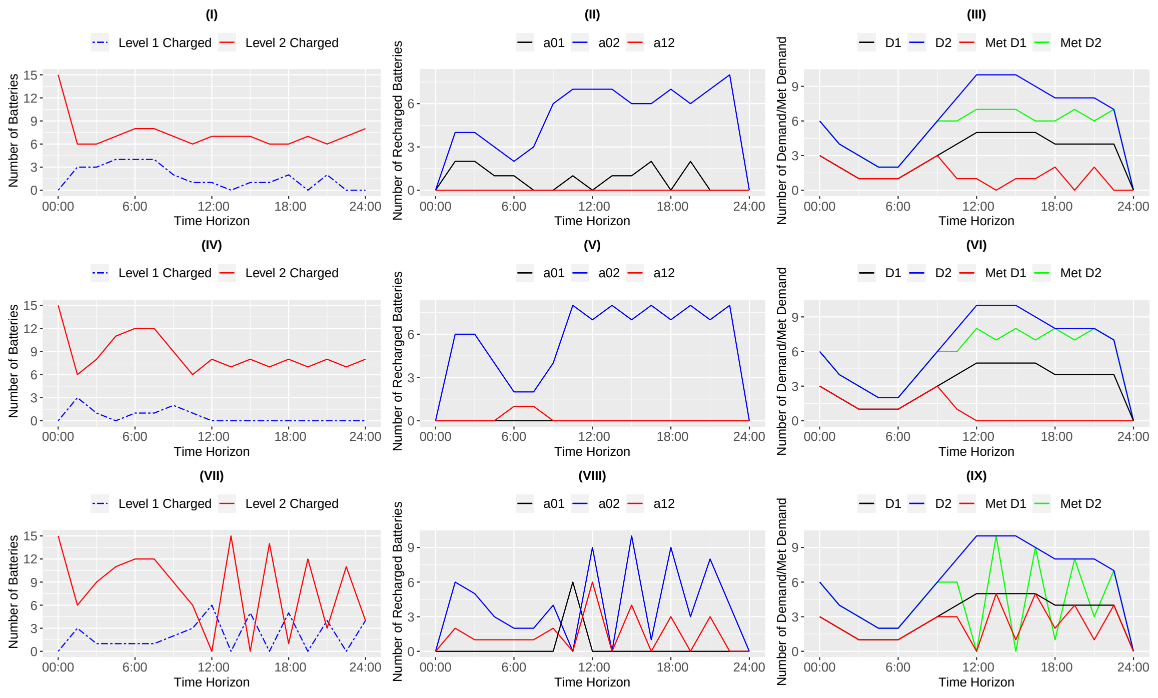


Figure 4.6: Sample paths of states (I, IV, VI), optimal policies (II, V, VII), and demands (III, VI, IX) for $\rho^{21} = 0.5, 1, 2$ when the realized demand of either class equal mean demand.

We also compare the average amount of met demand of either class and the optimal policies over time when ρ^{21} (controlled parameter by the station) varies between 0.5 to 2 with 0.1 increments. We calculate the percentage of demand satisfaction of either class by inputting the associated value of each class into Equations (4.21) and (4.22) for different values of ρ^{21}. We use Equation (4.24) to find the average number of actions over time over all sample paths. We summarized the result based on 500 sample paths of realized demand in Table 4.5. Intuitively, as ρ^{21} increased, more/less class 1 demand is satisfied using drones with level 2/level 1 charged batteries. On average, more recharging occurs from level 1 to level 2 to satisfy the demand of either class. For smaller values of ρ^{21}, increasing the parameter value provides more incentive for recharging more drones up to level 2 and satisfying both demand levels. However, for larger values of ρ^{21}, as level 2 charged batteries are used more to satisfy class 1 demand, fewer level 2 charged drones are available to satisfy class 2 demand.

$$\text{Average Number of Action } a^{01}, a^{02}, \text{ and } a^{12} = \frac{\text{Total Number of the Action over Time over Sample Paths}}{\text{Total Number of Sample Paths}} * 100\%. \quad (4.24)$$

Table 4.5: Average met demand and policies over time for 500 sample paths for different values of ρ^{21}.

ρ^{21}	Avg Met Dem. C1 with L1 Charge (%)	Avg Met Dem. C1 with L2 Charge (%)	Avg Met Dem. C1(%)	Avg Met Dem. C2(%)	Avg Met Both Class (%)	Avg a_{01}	Avg a_{02}	Avg a_{12}
0.5	30.0	13.8	43.6	74.5	63.6	0.67	4.97	0.05
0.6	25.7	16.2	41.9	76.2	64.2	0.45	5.14	0.06
0.7	23.8	17.6	41.4	76.8	64.4	0.35	5.21	0.08
0.8	22.3	18.9	41.3	77.1	64.5	0.27	5.25	0.09
0.9	21.0	20.0	41.1	77.1	64.5	0.20	5.29	0.11
1.0	15.9	24.9	40.9	77.5	64.6	0.00	5.32	0.27
1.1	13.9	27.0	41.9	77.5	64.6	0.00	5.24	0.41
1.2	13.1	27.6	40.7	77.5	64.5	0.00	5.22	0.46
1.3	12.6	28.0	40.7	77.5	64.5	0.00	5.20	0.49
1.4	12.2	28.5	40.7	77.3	64.4	0.00	5.17	0.53
1.5	11.5	29.7	41.3	76.6	64.1	0.02	5.08	0.62
1.6	12.4	33.4	45.8	73.4	63.7	0.15	4.76	0.84
1.7	12.5	36.4	48.9	70.4	62.9	0.25	4.48	1.04
1.8	11.1	38.1	49.2	69.7	62.5	0.28	4.34	1.17
1.9	10.3	39.2	49.5	69.0	62.1	0.31	4.24	1.26
2.0	9.1	40.4	49.5	68.6	61.9	0.32	4.15	1.35

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.

4.5.2.2 Analysis on the Number of Required Batteries

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 the swap station. First, we note that backward induction (BI) can solve the problem with at most 21 batteries 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 4.4 using BI and RL. For $M > 21$, we report the results of our RL method in Table 4.6.

Table 4.6: Computational time, memory used, and average percentage of met demand over time for 500 sample paths when $\rho^{21} = 0.5$ using the RL method.

M	Avg. Met Dem. (%)	Comput. Time (s)	Memory Used (GB)
21	76.3	599.2	13.4
22	76.9	654.1	13.4
23	81.0	690.1	13.4
24	82.0	705.2	13.4
25	84.5	765.1	13.4
26	86.7	781.4	13.4
27	87.3	935.2	13.4
28	88.5	1036.8	13.4
29	91.2	1037.8	13.4
30	91.7	1221.1	13.4
31	91.8	1298.3	13.4
32	94.2	1330.8	13.4
33	94.2	1337.3	13.4
34	94.9	1374.1	13.4
35	94.9	1384.5	13.4
36	95.6	1406.9	13.4
37	95.7	1426.6	13.4
38	95.8	1453.5	13.4
39	95.9	1633.4	13.4
40	96.1	1651.4	13.4
41	96.6	1703.0	13.4
42	97.4	1761.6	13.4
43	97.8	1819.7	13.4
44	98.3	1835.9	13.4
45	98.6	1969.3	13.4
46	98.7	2106.8	13.4
47	99.6	2227.0	13.4
48	99.6	2582.8	13.4
49	99.7	2717.7	13.4
50	99.7	2976.0	13.4
51	99.9	2984.6	13.4
52	99.9	3166.9	13.4
53	99.9	3169.9	13.4
54	100.0	3289.2	13.4
55	100.0	3314.5	13.4
56	100.0	3379.8	13.4
57	100.0	3431.3	13.4
58	100.0	3489.0	13.4
59	100.0	3593.0	13.4
60	100.0	3597.5	13.4

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 mean demand in Figure 4.7. As all batteries are initially available with a level 2 charge, we recharge fewer drones in the early morning. Then, we recharge more batteries 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.

4.5.2.3 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

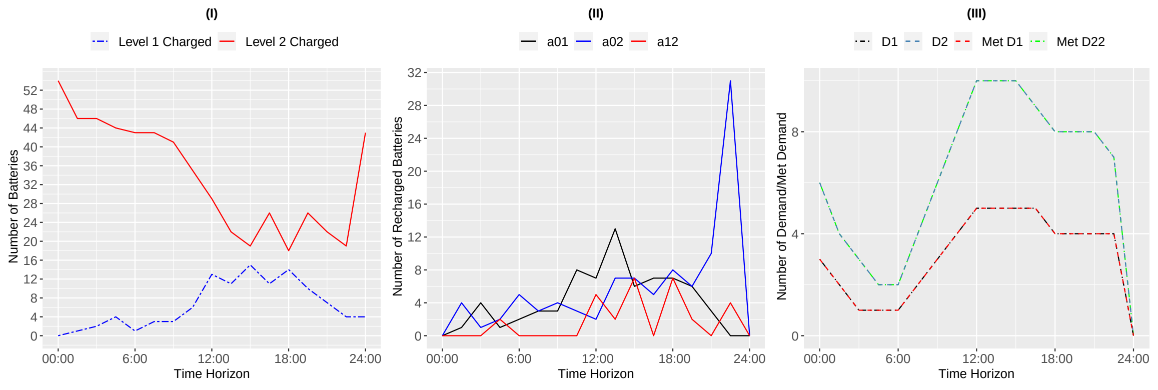


Figure 4.7: Sample paths of states, optimal policies, and demands for 54 drones, $\rho^{21} = 0.5$ when the realized demand of either class equal mean demand.

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 Figure 4.8, we show the average percentage of met demand for a different number of drones when we use/do not use demand classification. The blue 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. For this model, optimal policies indicate that more than 150 drones are needed to satisfy 100% of demand over 500 sample paths. In the model with demand classification, we note that finding the optimal policy and, in turn, the average percentage of demand for $M > 21$ is beyond our computational resources. Therefore, we only show the percentage derived from BI for $M \leq 21$ using the color red. However, using reinforcement learning (RL) enables us to offer near-optimal policies for this model. As shown, RL (black line in Figure 4.8) provides an upper bound for the optimal number of drones needed to satisfy a particular level of demand for the model with demand classification. As shown, RL's policies constantly outperform the optimal policy of the model with no demand classification in terms of the average percentage of met demand. For instance, RL's policies can satisfy 100% of the met demand with only 54 drones, which is significantly lower than the required number of batteries to hit this target when the demand is not classified.

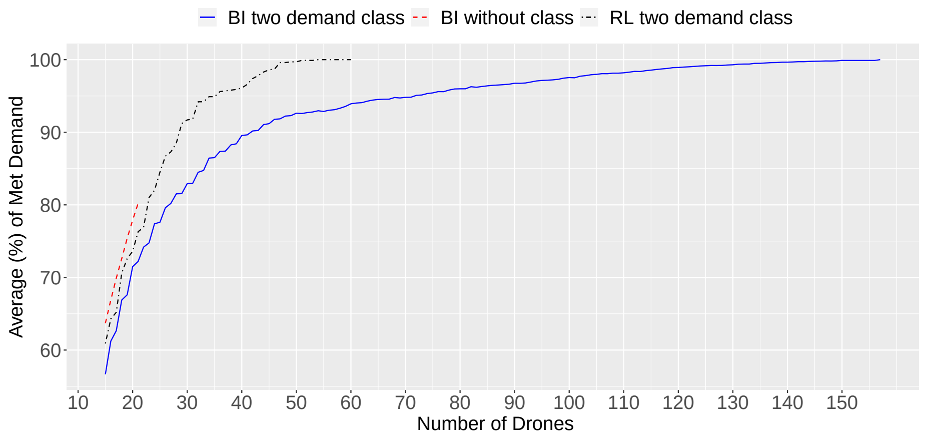


Figure 4.8: Average percentage of met demand for 500 sample paths when $\rho^{21} = 0.5$ using different models and solution methods.

4.6 Conclusion

In this research, we addressed managing 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 proposed stochastic scheduling and allocation problems 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. 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 observe that we run into time/memory issues when the number of drones is greater than 21. Hence, we applied a reinforcement learning (RL) method with an exploration feature to find high-quality approximate solutions quickly and overcome the time/memory issue. We designed a set of experiments to show the high performance of our RL 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.

We found plenty of directions and opportunities related to this work for future research.

For instance, in our work, there is no difference between the length of one level or two level recharging actions as we update the system's state every 90 minutes. Future research should consider the time difference between different recharging actions to capture the system's behavior more realistically. As our model is large-scale and complex, it is worth investigating the use of scalable solution methods, such as Q-learning, which are suitable for problems with no or incomplete models. In terms of modeling and applications, 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. It is also interesting to use the present model to manage the drone delivery system's operation for other applications and regions (e.g., Zipline drone delivery of the COVID-19 vaccine in Ghana). Another interesting research avenue is the incorporation of delivering multiple medical items with different demand distributions and/or demand classes using the same shared delivery resources. 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.

Acknowledgement

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.

Bibliography

Ackerman, E. (2020). Zipline launches long-distance drone delivery of COVID-19 supplies in the U.S. Last accessed on May 21, 2021 at https://spectrum.ieee.org/automaton/robotics/drones/zipline-long-distance-delivery-covid19-supplies.

Al-Sabban, W. H., Gonzalez, L. F., and Smith, R. N. (2013). Wind-energy based path planning for Unmanned Aerial Vehicles using Markov Decision Processes. In *2013 IEEE International Conference on Robotics and Automation*, pages 784–789.

Alagoz, O., Maillart, L. M., Schaefer, A. J., and Roberts, M. S. (2004). The optimal timing of living-donor liver transplantation. *Management Science*, 50(10):1420–1430.

Armony, M., Israelit, S., Mandelbaum, A., Marmor, Y. N., Tseytlin, Y., and Yom-Tov, G. B. (2015). On patient flow in hospitals: A data-based queueing-science perspective. *Stochastic Systems*, 5(1):146–194.

Baek, S. S., Kwon, H., Yoder, J. A., and Pack, D. (2013). Optimal path planning of a target-following fixed-wing UAV using sequential decision processes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2955–2962.

Bainbridge, R. (2019). Special report: Zipline international blood drone delivery service. Last accessed on May 21, 2021 at https://www.airmedandrescue.com/latest/long-read/special-report-zipline-international-blood-drone-delivery-service.

Baker, A. (2017). The American drones saving lives in Rwanda. Last accessed on May 21, 2021 at https://time.com/rwanda-drones-zipline/#:~:text=The%20drones%20can%20currently%20carry,platelets%2C%20plasma%20and%20juvenile%20blood.

Ball, M. (2020). Matternet unveils new medical drone payload exchange station. Unmanned Systems News. Last accessed on May 23, 2021 at https://www.unmannedsystemstechnology.com/2020/03/matternet-unveils-new-medical-drone-payload-exchange-station/.

Barmpounakis, E. N., Vlahogianni, E. I., and Golias, J. C. (2016). Unmanned Aerial Aircraft Systems for transportation engineering: Current practice and future challenges. *International Journal of Transportation Science and Technology*, 5(3):111–122.

BBC (2015). Electric scooter with swappable batteries hits market. Last accessed on May 23, 2021 at http://www.bbc.com/news/technology-33183031.

Benjaafar, S., ElHafsi, M., Lee, C. Y., and Zhou, W. (2011). Optimal control of an assembly system with multiple stages and multiple demand classes. *Operations Research*, 59(2):522–529.

Bertsimas, D. and Demir, R. (2002). An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science*, 48(4):550–565.

Chaharsooghi, S. K., Heydari, J., and Zegordi, S. H. (2008). A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45(4):949–959.

Chandler, S. (2020). Coronavirus delivers 'world's first' drone delivery service. Forbes. Last accessed on May 23, 2021 at https://www.forbes.com/sites/simonchandler/2020/04/03/coronavirus-delivers-worlds-first-drone-delivery-service/?sh=407e72774957.

Chang, Y. S. and Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104:307–317.

Chauvet, A. (2015). Ségolène royal: An agreement at COP21 would accelerate the energy transition. Last accessed on May 23, 2021 at http://www.20minutes.fr/planete/1742767-20151203-segolene-royal-accord-cop21-permettrait-accelerer-transition-energetique.

Chhatwal, J., Alagoz, O., and Burnside, E. S. (2010). Optimal breast biopsy decision-making based on mammographic features and demographic factors. *Operations Research*, 58(6):1577–1591.

Çimen, M. and Kirkbride, C. (2013). Approximate dynamic programming algorithms for multidimensional inventory optimization problems. *IFAC Proceedings Volumes*, 46(9):2015–2020.

Çimen, M. and Kirkbride, C. (2017). Approximate dynamic programming algorithms for multidimensional flexible production-inventory problems. *International Journal of Production Research*, 55(7):2034–2050.

Davitt, L. (2019). Long-range drones deliver medical supplies to remote areas of Malawi. Forbes. Last accessed on May 23, 2021 at https://www.forbes.com/sites/unicefusa/2019/06/19/long-range-drones-deliver-medical-supplies-to-remote-areas-of-malawi/?sh=259e1fd36add.

Dhingra, N. (2010). Estimate blood requirements - search for a global standard. Last accessed on May 21, 2021 at https://www.who.int/bloodsafety/transfusion_services/estimation_present.

DHL Press Release (2016). Successful Trial Integration of DHL Parcelcopter into Logistics Chain. Last accessed on June 7, 2021 at http://www.dhl.com/en/press/releases/releases_2016/all/parcel_ecommerce/successful_trial_integration_dhl_parcelcopter_logistics_chain.html.

Dhote, J. and Limbourg, S. (2020). Designing unmanned aerial vehicle networks for biological material transportation – The case of Brussels. *Computers and Industrial Engineering*, 148:106652.

Dongmei, L. (2016). China's BAIC group launches EV battery-swap station network in Beijing. China Money Network. Last accessed on May 23, 2021 at https://www.chinamoneynetwork.com/2016/11/03/chinas-baic-group-launches-ev-battery-swap-station-network-in-beijing.

Engineering for Change (2021). Special report: Zipline international blood drone delivery service. Last accessed on May 21, 2021 at https://www.engineeringforchange.org/solutions/product/zipline/#:~:text=Since%20launch%2C%20Zipline%20has%20over,arrive%20in%

20under%2030%20minutes.&text=The%20drones%20can%20deliver%20a,range%20of% 20about%2093%20miles.

Erdelyi, A. and Topaloglu, H. (2010). Approximate dynamic programming for dynamic capacity allocation with multiple priority levels. *IIE Transactions*, 43(2):129–142.

Federgruen, A. and Zipkin, P. (1984). Approximations of dynamic, multilocation production and inventory problems. *Management Science*, 30(1):69–84.

Fu, Y., Yu, X., and Zhang, Y. (2015). Sense and collision avoidance of unmanned aerial vehicles using markov decision process and flatness approach. In *Proceeding of the 2015 IEEE International Conference on Information and Automation*, pages 714–719.

FuelRod (2017). Power ready to go. Last accessed on May 23, 2021 at http://www.fuel-rod.com.

Fusheng, L. (2019). BJEV advocates battery swap service for e-vehicle owners. China Daily. Last accessed on May 23, 2021 at https://www.chinadaily.com.cn/a/201909/09/ WS5d75e342a310cf3e3556a816.html.

Gayon, J.-P., Benjaafar, S., and de Véricourt, F. (2009). Using imperfect advance demand information in production-inventory systems with multiple customer classes. *Manufacturing & Service Operations Management*, 11(1):128–143.

George, A. P. and Powell, W. B. (2006). Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65(1):167–198.

Gordon-Bloomfield, N. (2014). Forget better place: Simple battery swap technology from Slovakia proves its worth. Last accessed on May 23, 2021 at https://transportevolved.com/2014/ 02/12/forget-better-place-simple-battery-swap-technology-from-slovakia-proves-its-worth/.

Green, L., Kolesar, P., and Whitt, W. (2007). Coping with time-varying demand when setting staffing requirements for a service system. *Production and Operations Management*, 16:13–39.

Guerriero, F., Surace, R., Loscrí, V., and Natalizio, E. (2014). A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling*, 38:839–852.

Iliza, A. (2020). How can Rwanda tackle its blood donation shortfall? Last accessed on May 21, 2021 at https://www.newtimes.co.rw/news/rwanda-tackle-its-blood-donation-shortfall.

Jensen, J. (2019). Agricultural drones: How drones are revolutionizing agriculture and how to break into this booming market. UAV Coach. Last accessed on May 21, 2021 at: https: //uavcoach.com/agricultural-drones/.

Jiang, C. and Sheng, Z. (2009). Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications*, 36:6520–6526.

Jones, S. S., Stat, M., and Allen, T. (2007). Department Cycle to Improve Patient Flow. *Radiology*, 33(5):247–255.

Kent, C. (2019). Moving medical supplies: enter the drone. Medical Service Network. Last accessed on May 23, 2021 at https://www.medicaldevice-network.com/features/medical-supply-drones/.

Khojandi, A., Maillart, L. M., Prokopyev, O. A., Roberts, M. S., Brown, T., and Barrington, W. W. (2014). Optimal implantable cardioverter defibrillator (ICD) generator replacement. *INFORMS Journal on Computing*, 26(3):599–615.

Khoufi, I., Laouiti, A., and Adjih, C. (2019). A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles. *Drones*, 3(3):1–30.

Kwizera, O. and Nurre, S. G. (2018). Using drones for delivery: A two-level integrated inventory problem with battery degradation and swap stations. In *Proceedings of the Industrial and Systems Engineering Research Conferences*, pages 1–6, Orlando, FL.

Kwon, I.-H., Kim, C. O., Jun, J., and Lee, J. H. (2008). Case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications*, 35(1):389–397.

Lacey, G., Jiang, T., Putrus, G., and Kotter, R. (2013). The effect of cycling on the state of health of the electric vehicle battery. In *48th International Universities' Power Engineering Conference*, pages 1–7, Dublin, Ireland.

Lambert, F. (2018). NIO deploys 18 battery swap stations covering 2,000+ km expressway. electrek. Last accessed on May 23, 2021 at https://electrek.co/2018/11/15/nio-battery-swap-stations-network/.

Lyons, K. (2020). Zipline and Walmart to launch drone deliveries of health and wellness products. Last accessed on May 23, 2021 at https://www.theverge.com/2020/9/14/21435019/zipline-walmart-drone-deliveries-healthcare-amazon.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., and Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762.

Matternet (2020). Matternet's M2 drone system enabling new U.S. hospital delivery network at Wake Forest Baptist Health. Last accessed on May 23, 2021 at https://www.suasnews.com/2020/07/matternets-m2-drone-system-enabling-new-u-s-hospital-delivery-network-at-wake-forest-baptist-health/.

Maxwell, M. S., Restrepo, M., Henderson, S. G., and Topaloglu, H. (2010). Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281.

McNabb, H. (2020). A drone battery that charges in 5 minutes. Last accessed on May 21, 2021 at https://dronelife.com/2020/08/01/a-drone-battery-that-charges-in-5-minutes/#:~:text=Drone%20battery%20endurance%20is%20a,to%20charge%20the%20drone%20battery.

McNabb, M. (2020). Drone delivery in the era of the pandemic: From the floor at commercial UAV expo. drone life. Last accessed on May 23, 2021 at https://dronelife.com/2020/09/16/drone-delivery-in-the-era-of-the-pandemic-from-the-floor-at-commercial-uav-expo/.

Mlinar, T. and Chevalier, P. (2016). Dynamic admission control for two customer classes with stochastic demands and strict due dates. *International Journal of Production Research*, 54(20):6156–6173.

Mutzabaugh, B. (2017). Drone taxis? Dubai plans roll out of self-flying pods. Last accessed on May 21, 2021 at https://www.usatoday.com/story/travel/flights/todayinthesky/2017/02/13/dubai-passenger-carrying-drones-could-flying-july/97850596/.

Nasrollahzadeh, A., Khademi, A., and Mayorga, M. E. (2018). Real-time ambulance dispatching and relocation. *Manufacturing and Service Operations Management*, 20(3):467–480.

Nurre, S. G., Bent, R., Pan, F., and Sharkey, T. C. (2014). Managing operations of plug-in hybrid electric vehicle (PHEV) exchange stations for use with a smart grid. *Energy Policy*, 67:364–377.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Peermohamed, A. (2017). Reva founder looks to make electric cars affordable in India. Business Standard. Last accessed on May 23, 2021 at http://www.business-standard.com/article/companies/reva-founder-looks-to-make-electric-cars-affordable-in-india-117041201118_1.html.

Petrova, M. and Kolodny, L. (2018). Zipline's new drone can deliver medical supplies at 79 miles per hour. Last accessed on May 21, 2021 at https://www.cnbc.com/2018/04/02/zipline-new-zip-2-drone-delivers-supplies-at-79-mph.html.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, New York, NY, USA, 2 edition.

Powell, W. B. and Topaloglu, H. (2005). Approximate dynamic programming for large-scale resource allocation problems. *INFORMS TutORials in Operations Research*, pages 123–147.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Hoboken, New Jersey, 1st edition.

Roy, B. V., Bertsekas, D., Lee, Y., and Tsitsiklis, J. (1997). A neuro-dynamic programming approach to retailer inventory management. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 4052–4057.

Rwanda civil aviation authority (2021). Unmanned aircraft operations in Rwanda unmanned aircraft operations in Rwanda. Last accessed on May 21, 2021 at https://caa.gov.rw/index.php?id=110.

Ryzhov, I. O., Mes, M. R. K., Powell, W. B., and van den Berg, G. (2019). Bayesian exploration strategies for approximate dynamic programming. *Operations Research*, 67(1):198–214.

Savuran, H. and Karakaya, M. (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. *Soft Computing*, 20(7):2905–2920.

Shirk, M. and Wishart, J. (2015). Effects of electric vehicle fast charging on battery life and vehicle performance. In *SAE Technical Paper*, pages 1–13, Detroit, MI. SAE 2015 World Congress & Exhibition, SAE International.

Singh, I. (2021). Draganfly to start drone delivery of COVID-19 vaccine to rural Texas. Last accessed on May 28, 2021 at https://dronedj.com/2021/05/20/draganfly-drone-delivery-covid-19/.

Sinnott, R. (1984). Virtues of the Haversine. *Sky and Telescope*, 68(2):158.

Somarin, A. R., Chen, S., Asian, S., and Wang, D. Z. (2017). A heuristic stock allocation rule for repairable service parts. *International Journal of Production Economics*, 184:131–140.

Staedter, T. (2016). Drones now delivering life-saving blood in Rwanda. Last accessed on May 21, 2021 at https://www.seeker.com/drones-deliver-blood-rwanda-2045341414.html.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2 edition.

Swartzman, G. (1970). The patient arrival process in hospitals: statistical analysis. *Health services research*, 5(4):320–329.

Thompson, S., Nunez, M., Garfinkel, R., and Dean, M. D. (2009). Efficient short-term allocation and reallocation of patients to floors of a hospital during demand surges. *Operations Research*, 57(2):261–273.

Tiwari, Y., Goel, S., and Singh, A. (2014). Arrival time pattern and waiting time distribution of patients in the emergency outpatient department of a tertiary level health care institution of North India. *Journal of Emergencies, Trauma and Shock*, 7(3):160–165.

Tokekar, P., Vander Hook, J., Mulla, D., and Isler, V. (2013). Sensor planning for a symbiotic UAV and UGV system for precision agriculture. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5321–5326.

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

Vincent, J. (2021). Self-flying drones are helping speed deliveries of COVID-19 vaccines in Ghana. Last accessed on May 21, 2021 at https://www.theverge.com/2021/3/9/22320965/drone-delivery-vaccine-ghana-zipline-cold-chain-storage.

Wang, H., Zhao, D., Meng, Q., Ong, G. P., and Lee, D.-H. (2019). A four-step method for electric-vehicle charging facility deployment in a dense city: An empirical study in Singapore. *Transportation Research Part A: Policy and Practice*, 119:224–237.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, 11(4):679–697.

Weise, E. (2017). UPS tested launching a drone from a truck for deliveries. USA Today. Last accessed on May 21, 2021 at https://www.usatoday.com/story/tech/news/2017/02/21/ups-delivery-top-of-van-drone-workhorse/98057076/.

Widrick, R. S., Nurre, S. G., and Robbins, M. J. (2018). Optimal policies for the management of an electric vehicle battery swap station. *Transportation Science*, 52(1):59–79.

Zhang, J., Denton, B. T., Balasubramanian, H., Shah, N. D., and Inman, B. A. (2012). Optimization of prostate biopsy referral decisions. *Manufacturing & Service Operations Management*, 14(4):529–547.

5. Conclusions and Future Work

This dissertation contributes to the field of optimization under uncertainty wherein we propose novel models and solution methods for dynamic scheduling and allocation problems to manage the operations in Electric Vehicle (EV) and drone battery swap stations.

In Chapter 2, we presented a novel class of stochastic scheduling allocation and inventory replenishment problems (SAIRP) for managing internal operations in swap stations facing stochastic demands and non-stationary power prices. We consider the capacity of batteries, which impacts the drive/flight range of EV and drones in reality, and introduce battery degradation into the model. We formulated the problem using Markov Decision Process (MDP) model to determine the recharging, discharging, and replacement actions that maximize the expected total profit over time. Our MDP utilized an aggregated state by considering the average capacity of batteries instead of the individual capacity of batteries. We verified the benefits of the aggregation by comparing its solution with a disaggregated MDP and a Monte Carlo simulation. We discussed the scale of the problem, showed SAIRPs suffer from the curses of dimensionality, and solved the modest SAIRPs with backward induction. We developed a heuristic benchmark policy and a double pass reinforcement learning (RL) method with heuristic policy initialization (DHPI RL) as our approximate solution methods. Using the high-performance approximate solution methods, we provide policies for modest snd realistic-sized SAIRPs.

In Chapter 3, we focused on developing intelligent algorithms that exploit the structure of an optimal policy and value function of SAIRPs to overcome the curses of dimensionality. In this chapter, we showed that there exists a monotone non-increasing optimal policy in the second dimension of the state when there is an upper bound on the number of batteries replaced in each period. We also demonstrated that the MDP value function of the stochastic SAIRP is monotonically non-decreasing in the first, second, and both dimensions of the state. We exploited the monotonicity of value function to select the monotone approximate dynamic programming (MADP) as our solution approach. We analyzed the optimal policies of small SAIRPs and found the linear relationship between value and the model's parameters. Therefore, we developed our monotone approximate dynamic programming with regression-based initialization (MADP-RB), which is an enhancement of MADP, to initialize the algorithm intelligently. We designed sets of experiments and showed the high-performance of MADP-RB in solving modest and realistic-sized SAIRPs regarding the computational time, optimality gap, convergence, and expected total reward (profit). We provided insights for managing battery swap stations of modest and realistic-sized SAIRPs based on our analysis of results. A significant finding is that the optimal policy of modest size and realistic-sized problems can be significantly different. It further emphasizes solving realistic-sized problems using approximate solution methods. In other words, breaking down the problem into smaller pieces and solve them consecutively to optimality can not find the policies that can outperform near-optimal policies of realistic-sized problems solved by ADP/RL methods.

In Chapter 4, we proposed a novel class of stochastic scheduling and allocation problems with multiple classes of demand (SA-MCD) for delivering medical supplies using drones. We classified the stochastic demand for medical items based on the distance between the hospitals that requests delivery and the swap station located at the drone hub. We linked the classified demand to the charge inside the drones' batteries such that each class of demand can be satisfied using drones with a certain level of charge or higher. We sought to determine recharging actions that maximize the expected total weighted met demand. We fed realistic data from locations of the Zipline station, airports, and hospitals in Rwanda, the population of districts, flight regulations in Rwanda, and the Zipline drone configuration, including the speed, flight range, and recharging time. We applied an RL method with the exploration feature and showed its high performance compared to the backward induction solution. We provided our RL's solution for larger instances of SA-MCD that are not solvable using backward induction. We conducted different sets of experiments to obtain insights about the policies and the number of drones needed to satisfy a certain level of demand and performed sensitivity analysis on the parameter controlled by the station.

There are several directions to expand the work presented in Chapters 2 and 3. Regarding the model and solution methods, we suggest that future research should examine other state aggregation methods and analyze their impact on the computational efforts and quality of the solutions. Moreover, it is worthwhile to solve a disaggregated MDP with ADP/RL methods and compare the results with the presented aggregated model. It is also interesting to solve SAIRPs using other approximate solution methods such as Q-Learning and the policy gradient approach. In terms of battery swap station applications, researchers should examine utilizing different charging options, optimizing a swap station network, and analyzing the impact of varying the replacement threshold.

We suggest expanding Chapter 4 work in various directions. First, future research should consider the different lengths of recharging when levels of recharging differ. Given the complex transition probability function, we suggest applying a Q-Learning method that fits problems with no or incomplete MDP elements, such as transition probability when studying SA-MCD with more than two classes of demand. In this chapter, we provided demand classification based on the distance between locations. Future work can consider several classification criteria such as distance, level of emergencies, medical supplies, etc.