Degradation-based Optimal Swapping Policy

by

Ahmad Eyad Almuhtady

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2013

Doctoral Committee:

Professor Jun Ni, Chair
Erica Klampfl, Ford Motor Company
Post Doctoral Research Fellow Seungchul Lee
Professor Panos Papalambros
Professor Edwin Romeijn

ALL PRAISES TO ALLAH

To my mother;

without whom, I wouldn't have been able to traverse this journey.

ACKNOWLEDGEMENTS

I would like to thank Prof. Jun Ni, my committee chair, for his help and guidance. He has provided me with very valuable advice and direction; none of my work would have been possible without him. I would also like to thank Prof. Panos Papalambros, Prof. Edwin Romeijn, Dr. Erica Klampfl, and Dr. Seugchul Lee for serving on my doctoral committee and providing me with helpful suggestions on ways to improve my dissertation.

I would like to thank Dr. Michael Wynblatt for his help and suggestions in my research work. I would also like to thank Prof. Richard Scott and Prof. Kazu Saitou, from whom I have learned a lot. I would like to thank one of my best friends, Dr. Ali Nasir, from Aerospace Department for his help in the MDP implementation.

I am very grateful for my friends from the S.-M. Wu Manufacturing Research Center, especially Dr. Saumil Ambani, Dr. Adam Brzezinski, Xi Gu, Terri L. Gorowski, Dr. Xiaoning Jin, Dr. Shuhuai Lan, Prof. Lin Li, Xinran Liang, Xun Liu, Yangbing Lou, Dr. Jae-Wook Oh, Dr. Chaoye Pan, Xianli (George) Qiao, Prof. Mohammad Razfar, Dr. David A. Stephenson, Dr. Li-Jung (Bruce) Tai, Dr. Yong Wang, Xin Weng, Grace L.-H. Wu, Li Xu, Xinwei Xu, and Dr. Hao Yu. I would like to thank my friends in Ann Arbor who made my life here vibrant and exciting, especially Amjad Abu Jbara, Mousa Afaneh, Amr Alaa, Dr. Khaled Alashmouny, Mahmoud Alazzouni, Anas Aldomi, Khaled Aljanaideh, Ahmad Aljanaideh, Ahmed AlSabbagh M.D., Hisham Elmoaqet, Dr. Ahmad Hasan, Samer Kadous, Ahmed Mady, Professor Ahmad Nawafleh, and Professor Ahmad Safi. I share with you the best moments I

lived in this city.

Finally, I would like to thank my wonderful and supportive family. Dad, you have worked so hard to provide for me and allow me the chance to earn this degree. Mom, you have supported me through every single hardship during this journey. My sisters Dalia and Danna, and my brother Hashem, words cannot express my gratitude for you. I hope I can someday support others as well as you have supported me.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

CHAPTER I

Introduction

1.1 Background and Motivation

With usage and age, degradation is an inevitable course of any asset (tool, machine or system). The degradation of the asset's health state is met with proper maintenance actions that restore the health state to the point where the functionality of the asset is resumed. Maintenance has received significant attention in the last few decades in various applications. The appreciation of maintenance associated costs and the asset maintenance complexities due to technological advances have accelerated the evolution of the maintenance paradigm. For example, Koren [1] reported that maintenance was in fact the most important factor in manufacturing system cost. Mobley [2] reported that ineffective maintenance management would bring wasted maintenance cost of about $60 billion annually.

In one direction, maintenance paradigm evolved towards involving more prediction in the decision making. For example, in manufacturing it is easy to recognize the evolution from reactive maintenance (fail and fix), to preventive maintenance (timely-based maintenance), then to condition-based maintenance (monitor and diagnose), and finally arriving at the prognosis and health management (PHM) (predict and prevent). In another direction, maintenance plans have been increasingly integrated with the other asset management operations, such that it guarantees effective uninter-

ruptable or strategically interrupted optimal operations. Examples of those are joint production and maintenance planning in manufacturing and joint fleet management and maintenance in transportation.

In many applications, we are managing a system of identical assets or components utilized differently. The difference in utilization is caused by performing different operations or tasks, being under different loading conditions or being differently used in terms of frequency. Examples of these are ground fleets (public transportation busses, delivery trucks in shipping companies, etc.), airline fleets, group of identical machines in a factory or machine shop, and electrical and mechanical systems in residential and commercial complexes (elevators, HVAC units, water pumps, etc.). The assets definition here can extend as well for humans as Human Resources management. Examples of identical assets performing different functionalities or undergoing different loading profiles are medical personnel in hospitals and soldiers in battle fields. The former one undergoes different loading conditions as the work shifts vary (day and night shifts). The latter one undergoes different loading conditions per the deployment location.

The difference in use will generally result a difference in the degradation rate of these assets health states (see Figure 1.1). With time, the difference in the health states becomes noticeable, and the componenets reach the threshold at which a maintenance action is required. The maintenance action can be in the form of substitution (replacement with a new one), or repair. This maintenance action can be costly especially in the former case; additionally it might interrupt the operation and functionality of the asset at crucial times.

In many cases the system is only intended to be functional for a finite time horizon. For example, NYC Transit buses have an average life expectancy of 12-15 years. When their time on the street comes to an end, they are sold for their recyclable scrap value [3]. For such scenario and using a direct approach where each part is assigned the same operation, task, or loading profile throughout the system lifetime, parts of the

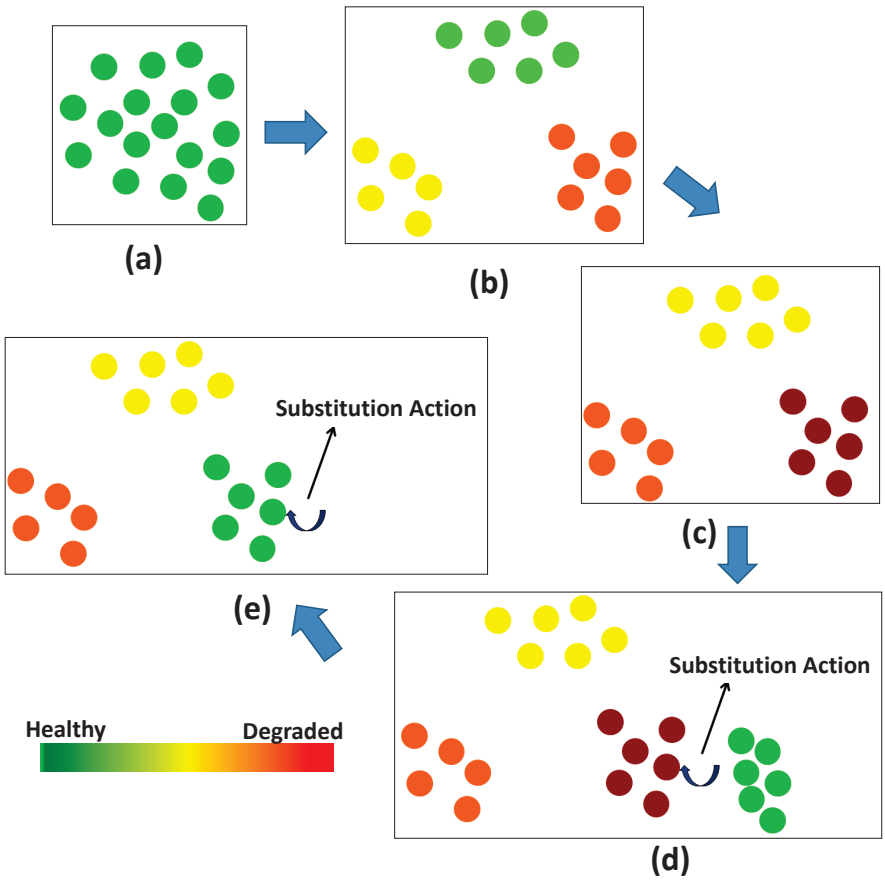


Figure 1.1: Health State Degradation in a System of Identical Components Utilized Differently.

system might undergo expensive maintenance actions (such as substitutions) just a short time prior to the retirement of the system. This is primarily due to the necessity to maintain the entire system functionality. However, with such occurrence the system is obviously under-utilized.

We define a new generic concept in joint asset management-maintenance denoted as Degradation-based Swapping. The degradation-based swapping relies on understanding the degradation evolution in the health states of the assets upon which it performs swapping actions that will promote better utilization of the system. A swapping action (see Figure 1.2) is defined as the inter-placement of two identical components operating under different loading profiles or have different rate of usages

within a system. The expensive maintenance actions are still ineludible; nonetheless swapping has the potential of reducing them.

We use this concept as the key stone in building a uniquely formulated and unprecedented (to the extent of our knowledge) resource allocation policy that optimally identifies the swapping and substitution actions necessary for utmost system utilization. We denote this policy as Degradation-based Optimal Swapping (DBOS) Policy.

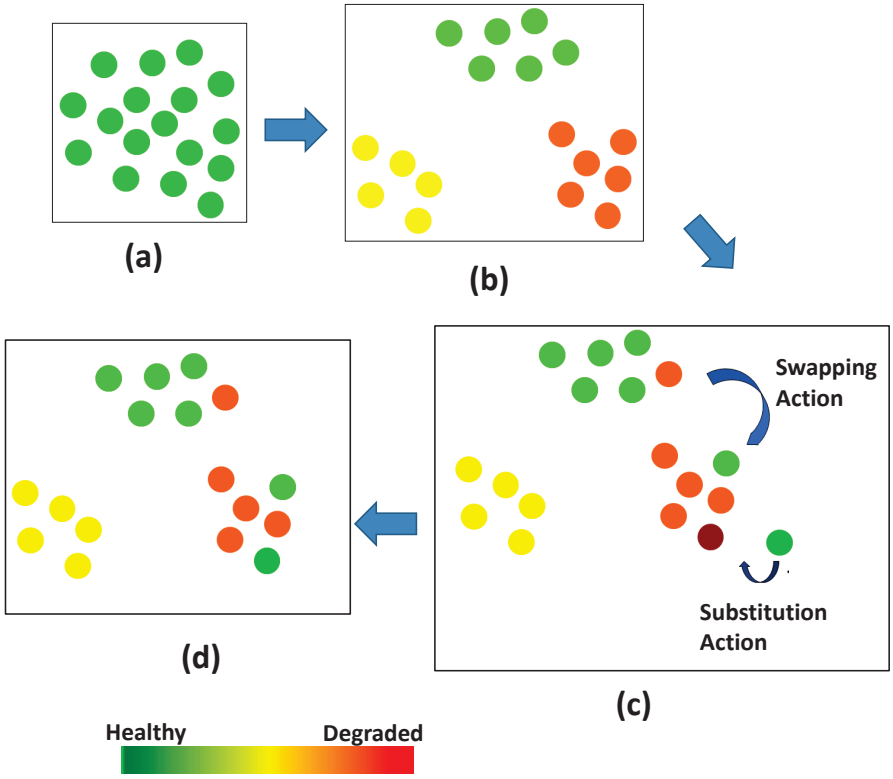


Figure 1.2: Degradation-based Swapping Anatomy.

1.2 Prospect Applications for DBOS and Relevant Research Work

The degradation based swapping concept is effectively generic that it can be applied in various disciplines. The DBOS policy can be used to utilize any assets as

long as the following conditions are met:

- A system of identical components or assets performing different functions.
- Degradation is correlated somehow to loading profile or frequency of use, and the degradation rates are sufficiently sparse(scattered).
- Swapping is much cheaper than substitution.
- Maintenance is conducted at pre-determined points.
- System is intended to be functional for a finite time horizon and will be retired after that.

We will demonstrate the output when some of these conditions are not met in Chapter II. Specifically, we will perform parametric case studies (see Section 2.4.2 that will illustrate favoring substitution over swapping when swapping and substitution costs come close to each other, or when the degradation rates are not sufficiently sparse.

Despite the requirement of attaining all the previously mentioned conditions for DBOS to promote utilization, we will demonstrate numerous fields and disciplines at which these conditions are met. We explain some applications in more detail (See Figure 1.3). Nonetheless, the applications that this concept can promote utilization in are countless. We take the chance also to review some of the closely related research work that has been done in the area of the application.

1.2.1 Fleet-level Battery Utilization for Electric or Hybrid-Electric Fleets (Transportation I)

While oil prices throughout the last decades have undergone significant increases, transportation still in general relies on it for 97% of its energy. Corporations, organizations, governmental agencies, and learning institutions are examples of fleet owners

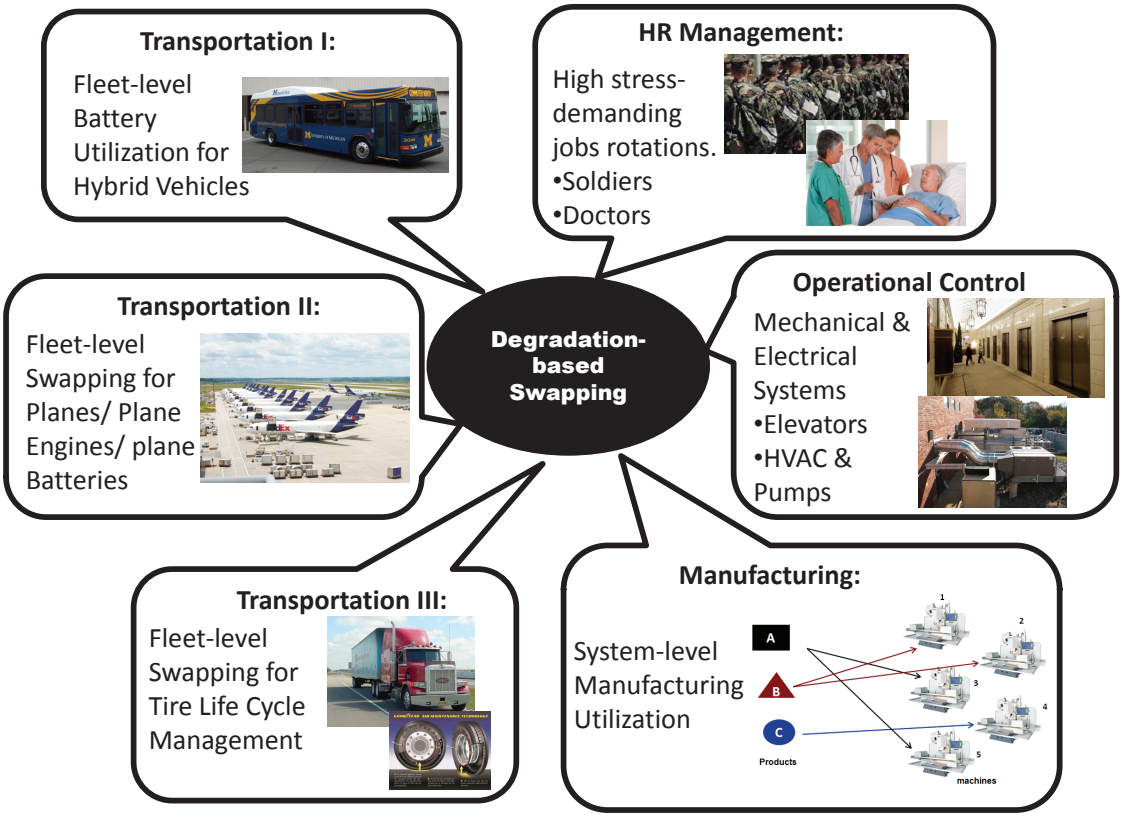


Figure 1.3: Prospect Application for Degradation-based Swapping.

who are significantly affected by that due to the amount of driving their fleets experience. For example, Walmart operated 7,000 trucks that in 2005 drove 872 million miles to make 900,000 deliveries to its 6,600 stores [4]. United States Postal Services (USPS) has 212,530 vehicles (letter carriers and trucks) which have driven a total of 1.3 billion miles in 2012 [5]. It becomes significantly harder for these companies and organizations with large fleets to maintain their preferred profit margins. Therefore, many of these fleet companies were highly motivated to reduce their annual fuel consumption which reflects on millions of dollars in savings by incorporating electric or hybrid electric vehicles in their fleets. Hybrid electric vehicles are those equipped with an internal combustion engine and a battery using both sources of energy appropriately, whereas electric vehicles depend solely on electricity, and hence acquire a much

more expensive battery on board. Class 6, 7 and 8 vehicles, especially in stop-and-go applications, are dominant with respect to fleet hybridization. Additionally, environmentally friendly technologies have attracted large companies and corporations who benefit from both commercial advertisement of endorsing such technologies, and established savings. One example of that is Walmart Corporation, which has set a goal of doubling the fleet efficiency by 2015 from a 2005 baseline, through a multistage plan that includes adding more electric and hybrid electric vehicles to the fleet. Both FedEx and UPS have as well endorsed hybridizing parts of their fleets by incorporating in their fleets 264 and 380 hybrid trucks, respectively [6, 7]. Examples of such plans are not inclusive to profit-motivated companies and corporations. They cover a wide and versatile spectrum that includes governmentally-managed departments, cities, and public and private schools and universities. Examples of cities who have already added or ordered hybrid buses for their public transportation systems include Washington, D.C. (950 hybrid buses), New York, NY (850 hybrid buses), Philadelphia, PA (480 hybrid buses), Minneapolis and St. Paul, MN (480 hybrid buses), Ann Arbor, MI, and Detroit, MI [8, 9]. Additionally, schools and universities (e.g. University of Michigan, MI and Kenton County School, KY) have as well introduced hybrid-electric buses into their fleets.

The hybrid systems are significantly costly. For example, the difference in cost between a standard public transportation bus and a hybrid one is more than $100,000 [8]. In a hybrid system, batteries have the most significant share of the total cost of the hybrid system [see Figure 1.4]. Lithium-ion batteries dominate the energy storage in hybrid systems by virtue of their high cell voltage, high energy density and excellent cyclability. However, Lithium-ion batteries unavoidably lose some capacity irreversible upon cycling. This capacity loss is often referred to as capacity fade or degradation. This degradation reaches a point where these batteries are no longer suitable for mobility application. This point is referred to as the "End of Life". The

United States Advanced Battery Consortium [10] defines the end of life for hybrid vehicle batteries to be the stage at which the battery meets specific failure criteria. The criteria state that failure occurs in the battery when the net delivered capacity of a cell, module, or battery is less than 80% of its rated capacity when measured on the Dynamic Stress Test (DST), or when the peak power capability (determined using the Peak Power Test) is less than 80% of the rated power at 80% depth of discharge (DOD).

At these batteries End-of-Life, substitution becomes inevitable. The substitution action here is defined as the replacement of the degraded battery with a new one. The limited battery useful life motivates the consideration of maintenance plans which can incorporate a predictive scheme of batteries health states evolution in the field. These plans have the potential to reduce the projected battery maintenance costs and can promote less abruptly interrupted daily task assignment to these hybrid vehicles through optimal utilization.

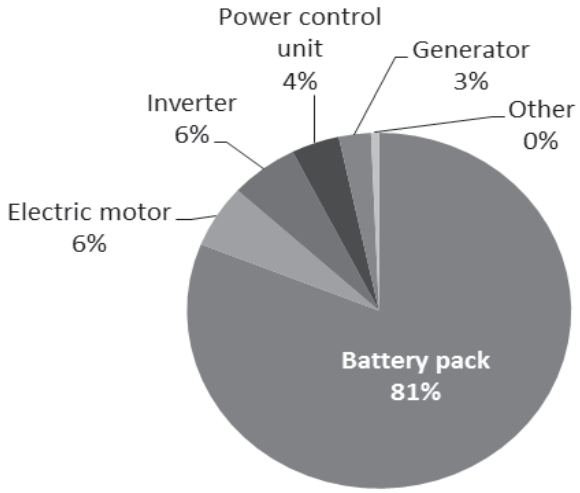


Figure 1.4: Breakdown of PHEV Drive System Cost by Component [11].

Fair prediction of the battery degradation within commercial fleets is attainable due to the consistency in the expected work load. For example, in a fleet of delivery trucks, the batteries in hybrid vehicles assigned to downtown area routes are

most likely subjected to larger frequencies of micro charging and discharging cycles in comparison to those within vehicles assigned to the suburban areas. Micro charging and discharging cycles refer to subsequent small increases and decreases of the battery charge. This reflects significantly on the degradation rate of these batteries. This consistency can help a predictive maintenance policy to optimally utilize all the batteries on fleet level.

We employ the degradation-based swapping concept to formulate such policy. Contrary to current practice where companies run batteries in the same loading profile until they reach retirement, the policy introduces swapping batteries within the fleet. The swapping action is defined here as the inter-change in the placement of two batteries from two different loading (degradation) profiles. Depending on the application, the swapping can include the whole vehicle or just the battery. In some cases the batteries are detachable from vehicle, such as in the case of some of the new designs for electric buses. In other cases, the batteries separation can be of difficult nature, and thus the whole vehicle would be moved to the different loading profile.

Relying on the prediction of the different degradation rates which is attributed mainly to the loading and usage conditions, the policy optimally places the batteries at each interval within the loading conditions that provide the best utilization. This will be shown to achieve a significant reduction in the projected cost of the maintenance plans. Additionally, the DBOS policy has the potential of providing an integration between maintenance actions and the company's daily operations (integration of maintenance and logistics). This enables a sustainable management of the costly hybrid fleet asset. Furthermore, the information obtained can be invested to build up a database of retired batteries in terms of their conditions and date of retirement. This database can significantly improve the success of the retired batteries remanufacturing schemes, already implemented (or under construction) in several OEMs. The remanufacturing helps both reduce the environmental impact resulting

from the disposal of such batteries and promote the use of cheap second-hand hybrid technologies.

In literature, three popular problems investigated with respect to ground vehicles fleet management are: the vehicle routing problem, the bus driver scheduling problem, and fleet replacement and lifecycle management. The vehicle routing problem aims to design a set of m minimum cost vehicle routes through n customer locations, so that each route starts and ends at a common location and some side constraints are satisfied [12]. The bus driver scheduling problem is involved with the assignment of drivers to a selection or working shifts satisfying the service requirements [13]. It is quite evident that these two problems are someway far from our objective. The fleet replacement and lifecycle management is in fact a series of problems that are involved primarily with the timely replacement of vehicles and equipment through prediction of asset lifecycles. A famous problem that focuses on this aspect is the bus engine replacement problem, which is also known as Zurcher's replacement problem [14]. The intended target was fleet maintenance management at the Madison (Wisconsin) Metropolitan Bus Company. Zurcher sought an age-dependent replacement policy to minimize expected total discounted or long-run average costs, where the cost function included the expensive replacement cost, and an age dependant monthly operating cost that incorporated unexpected failure cost component [15]. In a similar manner, DBOS will focus on minimizing maintenance costs of the fleet, which are mainly attributed to the replacement of batteries. The unexpected failure cost is introduced in DBOS as penalties incurred for over-usage. The main difference is that while Zurcher's problem focuses only on taking optimal replacement decisions, DBOS utilizes the fleet through the swapping actions, choosing optimal placements of the batteries within the degradation (loading) profiles, in addition to choosing optimal replacements (substitutions) to reduce the total maintenance plan costs.

We note that the fleet-level battery utilization will be the primary application

used in DBOS development in Chapters II through IV. Nonetheless, the described framework can be easily "tailored" to any of the applications mentioned in Figure 1.3 or other applications that can benefit from the degradation-based swapping concept.

1.2.2 Fleet-level Swapping for Planes/ Plane Engines/ Plane Batteries (Transportation II)

DBOS can be applied in airline fleets to utilize planes (or major components such as turbines and batteries) similarly as in ground vehicle fleets. However several conditions may require further modification to DBOS to make it applicable. The main reason behind this is the different locations the planes are at a specific time, rendering the swapping action more complex.

As it will be shown in Chapter II, DBOS model is expected to partially share the form of one of the most famous scheduling problems in airline fleet management which is globally known as the fleet assignment problem in transportation science. Given a flight schedule and a set of aircrafts of different types, the fleet assignment problem faced by an airline is to determine which type of aircraft should fly each flight segment on the airline's daily (or weekly) schedule [16]. The similarity between these two problems mainly arises in the placement decision variable; chosen to be binary in many cases, this variable holds the key to optimize the objective function [17]. In the fleet assignment problem, there are several factors considered in assigning a fleet to a flight leg. These factors include passenger demand, revenue, seating capacity, fuel costs, crew size, availability of maintenance at arrival and departure stations, gate availability, and aircraft noise. Many of these factors are captured in the objective coefficient of the decision variable; others are captured by constraints [18]. On a similar basis, modeling the problem for the DBOS policy is intended to take into account several factors, such as degradation profiles, demand, batteries health states tracking, maintenance capabilities and costs associated with the swapping and

substitution actions. However, there are several important differences between the two problems such as the substitution variables (reset variables) needed for DBOS to function properly. The substitution variables interaction with the placement variables and their major contribution in the objective function uniquely characterizes DBOS.

Airline planning process evolves through several decision making phases including schedule construction and fleet planning that are succeeded by aircraft maintenance routing and crew scheduling. The schedule is a list of flight numbers that gives the origin of the flight, its destination, time of departure, time of arrival, and days of the week that the flight operates [19]. The next step (the fleet assignment which is explained above) dictates which type of plane that will fly each flight leg. Hane et al. [18] and Abara [20] investigated, modeled and provided suitable optimization algorithms for this part. A third level of planning determines the actual routing of the tail numbers where maintenance considerations predominate. The maintenance routing problem has been considered in [20, 21, 22, 23, 19]. Initial research considered these steps of planning separately. Recently, the need for integrated planning and robust planning in this field was realized. Integrated planning is intended to integrate the functional phases at the planning stage, and robust planning is intended to make decisions at the planning stage that are beneficial to the operations [24]. Integrating schedule design and fleet assignment was implemented in [25, 26]. Examples of research on robust planning include robust fleet assignment as in [27, 28].

The swapping in airline fleet assignment has been sometimes referred to as the Re-fleeting problem, first introduced by Berge and Hopperstad [29]. The proposed concept, Demand Driven Dispatch (D^3), refers to the dynamic change of aircraft type assignments as the flight departure times approach and forecasts improve. The work mentioned restricts the change of assignment to one aircraft family due to the need to preserve crew schedules. Talluri [23] improved this swapping algorithm for a daily fleet assignment considering in specific the problem of changing the assignment of a spec-

ified flight leg to a different equipment type while still satisfying basic requirements (flow balance, flight coverage and equipment count). Acknowledged limitations to the mentioned work included the inconsideration of many operational constraints (maintenance and crew constraints) and restricting swaps between only two equipment types at a time. Jarrah et al. [30] improved and remodeled the re-fleeting problem as a multicommidity integer network flow problem with side constraints which allowed handling multiple aircraft types.

We make note here that while the re-fleeting problem represents swapping in principle, it is different from the degradation-based swapping concept in DBOS. The re-fleeting is concerned with swapping aircraft types (not tail numbers) and is not performed upon understanding of the health states evolution for optimum utilization of the fleet as DBOS does. The change in demand where feasible profitability is attainable, is the mere trigger in the re-fleeting problem. The reduction in the costly substitutions where feasible placement inter-changing is attainable, is the trigger in DBOS.

1.2.3 Fleet-level Swapping for Tire Life Cycle Management (Transportation III)

It does only take one trip on any US highway to recognize the significance these vehicles represent for the economy with their numbers. In fact there are almost half a million long-haul trucks on the road today [31] that haul over $8.3 trillion worth of merchandise annually [32]. Tires, like any component in a functional system, suffers degradation and requires maintenance from time to time. It was found that tire-related costs are the single largest maintenance item for commercial vehicle fleet operators with more than 50% of all truck and trailer breakdowns involving a tire in some way [33]. One of the unique characterizations of the tire degradation in long haul trucks is the tread wearout. Treads role is significant with respect to driving

performance and truck handling (safety). As a matter of fact, Federal Motor Carrier Safety Administration has regulations for a minimum legal tread groove pattern depth of at least 4/32 of an inch on any tire on the front wheels of a bus, truck, or truck tractor, and 2/32 of an inch on any other nonfront tires [34]. The uniqueness of tread wearout is the ability to restore the tread through a relatively complex process called retreading. While retread tires costs 30% to 50% in comparison to new ones, retreading can effectively return the tire to a "like new" condition if it is done correctly and at the right time. The success of this process decreases significantly when the tire is overused. Retreading can be done several times before the tire is sent for retirement. With this complex lifecycle, fleet companies and tire manufacturers are seeking life-cycle management policies to enable optimal utilization of the tires.

Degradation-based Swapping in tires life cycle management has been present for many years in a simple form. Tire manufacturers recommend rotating tires every 5,000 to 10,000 miles. This rotation is a form of a fixed swapping that is influenced by the fact that different tires within the same vehicle degrade differently. We will show in Chapter II that DBOS, being optimal, has the potential of outperforming this fixed swapping significantly. In the truck tire life cycle management, DBOS can be helpful if swapping is allowed. There can be two forms of swapping in this case: (1) on truck level where the position of the tire within an 18-wheel truck is significantly correlated to its degradation, and (2) on fleet level where different trips, loads, road conditions, etc. have direct correlation with the tread wearing out. The retreading event in this case will represent the reset maintenance action (like substitutions in fleet level battery utilization). We are currently working closely with a major tire manufacturing company, to assess DBOS ability of being part of a holistic long-haul truck tire life cycle management. The details of this complex management is beyond the scope of this thesis and is protected under a non disclosure agreement.

1.2.4 System-level Manufacturing Utilization

In many manufacturing systems and manufacturing job shops, there are several machines that are capable of performing a number of different processes required as part of the production process. In these cases, identical machines can be used to perform different processes or produce different products. We can link this group of identical machines performing similar or different tasks in a manufacturing system or manufacturing job shop to fleet operation, and will refer to it as fleeted manufacturing group.

The degradation in the health state of the key tool in these machines can be correlated to the task assigned or product produced. Therefore, the scheduling and assignment of these tasks for these machines will directly affect the anticipated maintenance actions. The number of machines within a fleeted manufacturing group dedicated to perform certain task, or produce certain product is solely dependent on the demand. However, the specific machines within this group chosen to perform a certain task, is a choice of the scheduling authority in the production plant as long as the demand is satisfied. Therefore, the scheduling can be"tailored" towards specific production and maintenance outcome, and the control over this can be established. This control can be of great significance for plans with finite time horizons. For example, the scheduling can be chosen to prevent any maintenance actions in specific production times in the plan horizon (e.g., critical production times, high demand intervals, etc.), or force the maintenance action to be taken with specific capabilities (e.g., maintenance crew is available in certain days, maintenance crew capability is limited, etc.). If the scheduling indicated above is capable of optimally utilizing the key tools of the machines within a fleet in the optimum feasible way, meanwhile satisfying the conditions stated above (demand, capabilities, etc.), the scheduling can become an optimum policy for production and savings can be maximized.

For further clarification, an example is shown in Figure (1.5a), where five identical

machines have the capability of producing any of the three products A, B, and C. The number of machines assigned to produce a specific product is solely dependent on the demand, and cannot be changed. However, the choice of assigning a specific machine to a specific product can be established as long as the demand is satisfied. In this example, at a specific instant, two machines, another two machines and one machine is assigned to products A, B, and C, respectively. The solid, dashed and dotted arrows represent for example the assigned tasks/products for each machine at 3 consecutive hours.

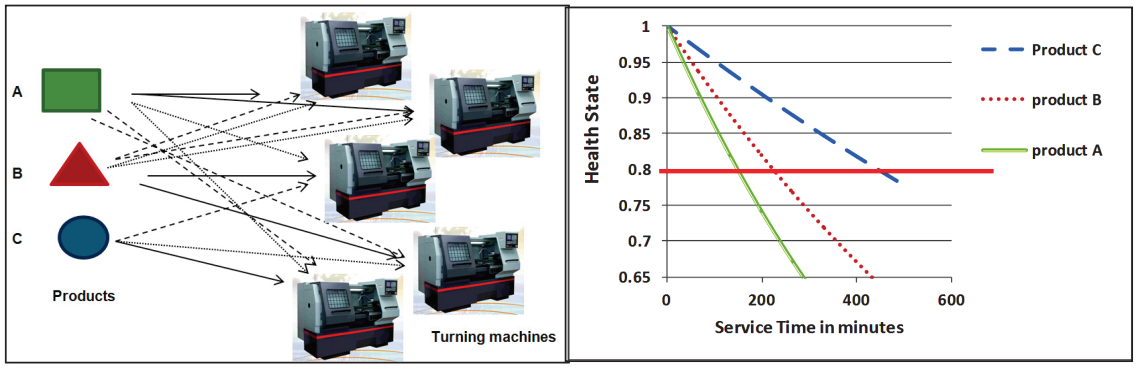


Figure 1.5: (a)An Example of a Fleet of Identical Machines Producing Different Products. (b)An Example of Machine's Key Tool Degradation Profiles When Producing Different Products.

In Figure 1.5b, an example of the different degradation profiles of the health state of the machine's key tool (blade) is shown. The horizontal line represents (for example) the threshold in the health state at which the key tool requires a prescribed maintenance action (e.g., replacement, re-shaving).

In most situations, scheduling in manufacturing systems is running independently of the understanding of the key tools health state degradation. The tasks are assigned to the fleeted manufacturing group either in a fixed manner, promoted by dedicated manufacturing scheduling, or randomly assigned. However, DBOS can provide better utilization through swapping assignments/products amongst the group [35, 36]. This

can achieve a reduced and/or controlled maintenance actions as key tools' remaining useful life is effectively utilized. The policy will primarily rely on the prediction of the different degradation rates which are attributed mainly to the loading and usage conditions (tasks/products). The prediction of such degradation level introduces a potential to conduct swapping actions of these products/tasks amongst the different machines, enabling the control of the end of life for these machines key tools. One direct impact is in the form of providing significant savings in projected maintenance costs for finite time horizon plans, when such policy is applied. Additionally, this policy provides the ability to conveniently incorporate maintenance actions with the company's daily operations.

In the last decades, maintenance scheduling has received significant focus in literature as the paradigm of maintenance has been shifting from complete dependence on the age-dependent preventive maintenance (PM) policies [37, 38, 39] to condition-based maintenance as in [40, 41, 42]. In most of these research efforts, production scheduling has not been the focus and is assumed to be decided independently from the maintenance anticipated actions. Yang et al. [43] proposed a new method for scheduling of maintenance operations in a manufacturing system using the continuous assessment and prediction of the level of performance degradation of manufacturing equipment, as well as the complex interaction between the production process and maintenance operations. The cost effects of different maintenance schedules were assessed and an optimum maintenance scheduling has been chosen utilizing Genetic Algorithm (GA).

One of the first efforts investigating joint maintenance-production scheduling policies is found in [44]. When the restoration cost function has a linear or exponential form, the mentioned work was able to find optimal simultaneous determination of the number of equal-interval maintenance inspections in a production run, the length of the production run and consequently the economic manufacturing quantity, and

the maximum level of backorders. Other efforts in joint maintenance production scheduling are found in [45, 46, 47, 48, 49].

In this application, DBOS uniqueness arises from the fact that it is intended to "tailor" the production towards some maintenance outcome and deals with fleets of identical machines working on similar or different assignments or products.

1.2.5 Other Prospect Applications for DBOS

The DBOS concept is very generic and is applicable to numerous fields. For all the previously mentioned applications, DBOS or some sort of swapping has been applied to. However, there are other applications to which DBOS has prospect of being applied. DBOS can be used to promote utilization in residential and commercial complexes taking advantage of the after hours partial shut down of some of the mechanical and electrical systems (See Section 5.3.1.1). With advances in psychology science, DBOS has the potential to help in Human Resources (HR) management especially pertaining disciplines where individuality is almost nonexistent (See Section 5.3.1.2). DBOS has also the potential to be applied in other transportation fields such as swapping ships and fleets (based on area of operation) and swapping railroad cars in rail transport system.

1.2.6 DBOS Analogy to Sports

We conclude this section pointing the DBOS policy analogy to sports. The policy mimics the mentality of the team coach in sports. For example, in soccer, there is limited number of substitutions allowed for each team. Additionally, the team members on the bench are not as good as the ones on field. This could be analogous to saying that a substitution will be costly. Additionally, the different players on the field get tired at different rates; mainly based on the position of the player. For example, a midfielder covers twice as much distance per match as some defenders.

This is again analogous to the different degradation rates in the assets within the system based on the loading conditions under which they operate. The analogy of the proposed policy to sports can be seen clearly in what is defined as modern soccer. In modern soccer, the players are loosely positioned in the field in comparison to old traditional soccer [see Figure 1.6]. Strikers may get back sometimes and play as midfielders; defenders can go forward and assist in attacks and vice versa. A successful coach will be the one who teaches his players through training how to implement these swapping actions in position on an optimal level. This decreases the necessity to substitute as players will share the responsibility across all positions. Substitution takes place only when a player is completely exhausted. In the same sense, DBOS policy will attempt to swap the assets amongst different loading conditions, therefore sharing the degradation across the system. Substitutions are only done when reaching degradation threshold.

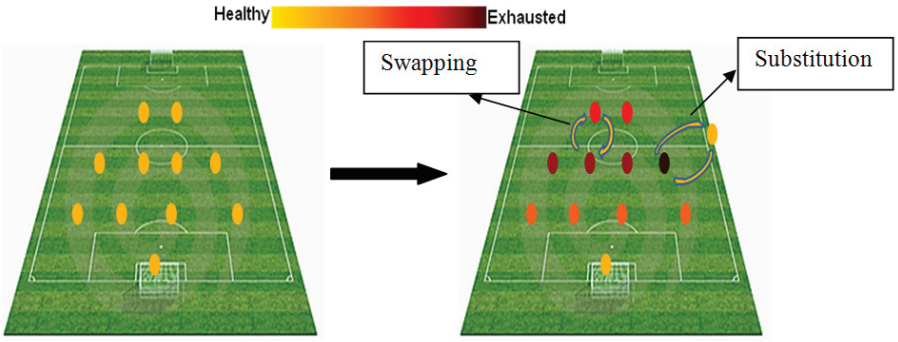


Figure 1.6: DBOS Policy Analogy to Sports.

1.3 Research Objectives

After we have introduced the degradation-based swapping concept and its various prospect applications, the research presented in the remainder of this thesis has three main objectives.

First, we want to transfer the Degradation-based Optimal Swapping concept to a representative concise mathematical model. We start with deterministic degradation estimation where the policy in this case will be in the form of an optimal schedule. Using the primary application in this thesis, the schedule should provide comparable or better performance than existing fleet management policies in terms of maintenance plan projected costs and fleet utilization. The optimization of the generated model will be investigated through standard optimization algorithms and will include the development of a swapping-specific optimization algorithm to robustly and repeatedly acquire global optimal solutions.

Second, we want to augment this model to account for uncertain degradation. This augmentation will be in the framework of the Stochastic Dynamic Programming (SDP) and Markov Decision Processes (MDP). This augmented model should be able to generate a policy that is capable of adapting to uncertain degradation, and hence is expected to provide robust performance with respect to uncertain degradation.

Third, we want to integrate a local inventory management model to the policy. The inventory model should account for inventory deterioration as the primary application in this thesis (batteries) have been shown to demonstrate such phenomenon. This inventory integration should be able to generate optimal replenishment policies and handle special instances such as replenishment with lead time, and special pricing circumstances.

1.4 Outline of the Dissertation

The remainder of this dissertation is organized as follows.

Chapter II explains the modeling of DBOS policy and the optimization of the generated model. A deterministic degradation is adopted in this chapter to understand the capability, performance and dynamics of DBOS. Upon optimization, the generated model will establish a series of optimal placements and substitution actions that

promote enhanced utilization of the system. Experimentation on the optimization of various instances based on the primary application in this thesis is conducted with several standard optimization algorithms. The development of a swapping-specific branch and bound based optimization algorithm will be shown to have significance with respect to larger size problems. The performance of DBOS will be compared with other policies currently employed in practice. This chapter is based on work described in [17, 50] with respect to the fleet-level battery utilization application and [35, 36] with respect to the manufacturing system-level utilization application.

Chapter III focuses on augmenting the model developed in Chapter II, in order to account for uncertainty in the health state degradation estimation. The augmentation is conducted through the use of stochastic dynamic programming (SDP) framework built on Markov Decision Processes (MDP) evolution. The generated policies upon this framework are studied extensively, testing them against variation of uncertainty distributions and misinformation of uncertainty distribution. This chapter is based on work described in [51].

Chapter IV is concerned with the integration of a local inventory model with the stochastic DBOS policy. The generated model will represent an integrated DBOS-Inventory policy that is capable of achieving optimal replenishment of the new components (batteries in the primary application) needed for the substitution actions, associated with optimal management of the system. The local inventory model will account for the deterioration of the components while in inventory. Case studies will show how the integrated policy will only stock up (optimally) when inventory is motivated such as in cases relevant to replenishment with lead time, and cases with special pricing circumstances.

Finally, Chapter V presents conclusions, contributions, and future work.

CHAPTER II

Deterministic Degradation-based Optimal Swapping

2.1 Introduction

In the previous chapter we have established the generic degradation-based concept as a management tool for a system of identical components operating in different conditions. In this chapter, we present the formulation of the Degradation-based Optimal Swapping (DBOS) policy model, in which deterministic degradation will be assumed. Incorporating uncertainty adds more complexity to the problem. Therefore, it is widely found in literature that uncertainty is incorporated in later efforts. Grossmann [52] had efficiently summarized the motivation behind this in three points. Deterministic models help overcome complexities arising from the nature of the problem itself. Additionally, deterministic models can be used to analyze different scenarios for the uncertain parameters avoiding complex stochastic models. And finally, in many applications the deterministic modeling forms the basis for the stochastic one such as when Petkov and Maranas [53] extended the pre-formulated combined production planning and scheduling model first proposed by Birewar and Grossmann [54] by including demand uncertainties. In this case, we are following the third scenario where we aim to extend the Deterministic DBOS scheme that we will develop in

this chapter to account for uncertainty in the next chapter. This will enable us to comprehensively understand the dynamics of DBOS.

This chapter will include as well an investigation of suitable approaches to achieve the optimum solution for the generated model. The formulation will be based upon the primary application in this thesis which is the fleet-level battery utilization. However, the same equations can be used for some of the other applications directly, and might require small retrofitting in others.

This problem can be categorized under the planning and scheduling optimization. The output can be in the form of a schedule of different placements for the batteries within the fleet, with optimally selected substitution actions occurring from time to time. Both planning and scheduling deal with the allocation of available resources over time to perform a collection of tasks. The difference between planning and scheduling is not always clear cut. However, in general planning deals with longer time horizons (e.g., weeks, few months) and it deals with high level decisions such as investment in new facilities and production levels. Scheduling on the other hand is concerned with shorter time horizons (e.g., days, few weeks) with the emphasis often being on the detail level decisions such as sequencing of operations [52]. Although the expected outcome decisions from the DBOS policy are low level decisions such as the change of the loading profile at which the battery is placed, DBOS is intended to be part of a long maintenance plan horizon. Therefore the policy can be classified under either scheduling or planning. DBOS model is expected to partially share the form of one of the most famous scheduling problems which is globally known as the fleet assignment problem in transportation science. Given a flight schedule and a set of aircrafts of different types, the fleet assignment problem faced by an airline is to determine which type of aircraft should fly each flight segment on the airline's daily (or weekly) schedule [16]. The similarity between these two problems mainly arises in the placement decision variable; chosen to be binary in many cases; this variable

holds the key to optimize the objective function [17]. In the fleet assignment problem, there are several factors considered in assigning a fleet to a flight leg. These factors include passenger demand, revenue, seating capacity, fuel costs, crew size, availability of maintenance at arrival and departure stations, gate availability, and aircraft noise. Many of these factors are captured in the objective coefficient of the decision variable; others are captured by constraints [18]. On a similar basis, modeling the problem for the DBOS policy is intended to take into account several factors, such as degradation profiles, demand, batteries health states tracking, maintenance capabilities and costs associated with the swapping and substitution actions. However, there are several important differences between the two problems such as the substitution variables (reset variables) needed for DBOS to function properly. The substitution variables interaction with the placement variables and their major contribution in the objective function uniquely characterizes DBOS. The daily scheduling of the fleet assignment problem formulation imposes large number of integer variables and severely degenerate model which leads to poor performance of standard linear programming techniques. Methods to address this problem include an interior-point algorithm, dual steepest edge simplex, cost perturbation, model aggregation, branching on set-partitioning constraints, and prioritizing the order of branching [18].

Planning and scheduling problems generally incorporate discrete/continuous optimization problems. The mixed integer nonlinear program (MINLP), inherently requires special treatment as complexities arise due to nonlinearity and integer choices. The most common MINLPs encountered in planning are 0-1 integer nonlinear programming (ZOINLP) problems where none of the continuous variables exist and all the decision variables are binary (zero or one). Details of the modeling of DBOS policy revealed in Section 2.2 will show that the generated model belongs to the (ZOINLP) problems category.

The basis of tackling integer programming problems (whether linear or nonlinear)

in many algorithms relies on relaxing the problem into continuous sub-problems. The algorithm in this case works on a higher level establishing control on the sub-solvers and using the information from the sub-problems solutions to arrive to the integer solution. The sub-problems are solved by some well-performing continuous variable programming problem solver (such as Simplex for linear programming (LP) problems [55] and Sequential Quadratic Programming (SQP) with reduced gradient method [56] for nonlinear programming (NLP) problems). Branch and Bound (B&B) algorithm by [57] falls under this category of integer programming problem solvers. B&B consists of a tree enumeration in which LP or NLP sub-problems are solved at each node, and eliminated based on bounding properties. B&B's success and speed in finding the solution inherently depends on the relaxed problem sub-solver.

Other algorithms for solving MINLP include Generalized Benders Decomposition (GBD) [58, 59], and Outer-Approximation (OA) [60, 61]. These are iterative methods that solve a sequence of alternate NLP sub-problems with all the zero-one variables fixed, and Mixed Integer Linear Programming (MILP) master problems that predict lower bounds and new values for the zero-one variables [52]. The LP/NLP based branch and bound in [62] integrates both subproblems within one tree search. The Extended Cutting Plane method (ECP) by Westerlund and Pettersson [63] is not involved with the solution of the NLP sub-problems, and rather uses successive linearizations. All these methods assume convexity to guarantee convergence to the global optimum. Literature also provides some non-rigorous methods for handling non-convexities such as the equality relaxation algorithm [64] and the augmented penalty version of it by Viswanathan and Grossmann [65].

Recently, stochastic methods have gained popularity over most conventional calculus-based search algorithms. This increased popularity is attributed to the successful implementation of these algorithms to solve various optimization problems which contains variables with discrete choices such as integer, binary, discrete set, etc. Among

these methods, genetic algorithms (GAs) and simulated annealing (SA) have been widely studied.

GAs were originally developed by [66] based on the Darwinian theory of biological evolution. Hwang and He [67] summarized GA's advantages over some conventional calculus-based search algorithms. First, GA imposes no limitation (such as continuity and differentiability) on the search space of the optimization problem. Secondly, a GA searches for the optimum solutions by parallel processing a population of solutions rather than just a single solution. Thirdly, a GA is based on natural selection criteria rather than deterministic rules and its search procedure is based predominantly on genetic operations. Finally, the GA search process has no need for any mathematical knowledge other than the fitness value of each potential solution. For the reasons described above, GA-based methods might have better chance in obtaining optimum (or near-optimum) solutions than calculus-based algorithms. The major disadvantages of GA includes the lack of assurance for obtaining a global optimum, lack of assurance of constant optimization response times, and incompatibility for online control applications due to the randomness in solutions and convergence.

There are many versions of GA. The simplest GA encodes all the individuals in the population into binary and performs the recombination on them, just before they are decoded back to real numbers in order to calculate the fitness in the next generation. This algorithm is known as binary genetic algorithm (BGA). The encoding and decoding in BGA causes the optimizer to spend a considerable time. Therefore, there has been an increase in the interest of GA's that can apply the recombination operators on real numbers with no encoding and decoding to binary. The GA developed for such purposes is called real parameter genetic algorithm (RGA). Other classifications of GA depend on whether the implementation is sequential or parallel. Sequential implementation represents the default GA where all of the steps explained above are done iteration after another on one processor. Typically this is used to

describe the generational sequential GA, while steady state sequential GA replaces only a portion of the current population with the new offsprings rather than creating an entirely new population. Parallel GA is related to the involvement of several processors to perform the steps at the same time. The difference between the centralized parallel GA and the distributed parallel GA [68] lies in the distribution of the roles for these processors. In centralized parallel GA, there exists a master processor which synchronizes the actions of the processors which perform the evaluations and recombination operations. In the distributed parallel GA, the master selection step is replaced by local selection routines which are distributed over the processors which already contain routines for evaluation and recombination.

Simulated annealing (SA), which is based on the physical process of annealing [69], is another widely applied stochastic method. When implemented successfully, SA shows good hill-climbing ability as it converges towards the optimal solution. Hence SA is considered one of the powerful tools for solving complicated problems such as combinational optimization problems. Because of the random nature of the search process used to identify the optimal solution, the convergence speed of SA is very slow [70]. However the merits of such algorithm are well acknowledged which motivated the inclusion of this algorithm in many hybrid GA algorithms as in [67, 71].

2.2 Mathematical Modeling of Deterministic DBOS Policy

The key to apply the DBOS policy is a concise and representative model which accounts for swapping and substitution actions. The objective of the policy aims towards optimal battery utilization over a finite plan horizon in a way that minimizes total maintenance plan projected costs.

Typical constraints are formulated for demand (number of vehicles operating in each degradation profile), batteries health state degradation tracking (swapping and substitution effects, threshold, etc.). Other constraints are relevant to the company's

logistics such as maintenance crew availability, business requirements, etc. The model includes two types of decision variables: placement variables and substitution variables.

2.2.1 Placement Decision Variables

The model is formulated to follow the placement of batteries in terms of location and time. The location here refers to the loading profile in which the battery is placed, and for which predicted degradation rate of the health state is assumed to be known. The variable is studied at predefined constant discrete intervals of time (Δ), which are chosen upon the company's preference and capability to achieve regular workflow. This interval should be inspired by the company's prescheduled checkups cycles. For example, if the company's vehicles are usually maintained or checked up monthly, then choosing Δ to be equal to 1 month is reasonable. Δ relates the frequency of the discrete time points at which the scheduler has the option to perform a swapping action.

Theoretically as Δ gets smaller, more swapping options are present, and we expect the total maintenance cost to decrease or remain the same. We will denote this rule as the DBOS-Discretization interval correspondence. The total maintenance cost remains the same and introducing further swapping actions will not improve the cost function, and the optimizer opts for no additional swapping actions upon correct implementation of the policy (accurate optimization). This DBOS rule is very important as it allows us to verify the robustness of optimization algorithms when handling DBOS instances. For a fixed plan horizon, as Δ decreases, the number of decision variables increases. This is related to the fact that there will be more points at which we have to make a decision resulting in more decision variables. The increase in the number of decision variables can be used to test the performance of different optimization algorithm (in terms of handling DBOS model). Specifically, we

determine the optimal cost for a given case study. Using the correspondence above, we have then established the lower bound. That means if decreasing Δ for the same plan horizon of this case study results in a cost higher than the one we found earlier, this will reveal the inability of the employed optimization algorithm to handle the DBOS model. This will be detailed in Section 2.4.2.

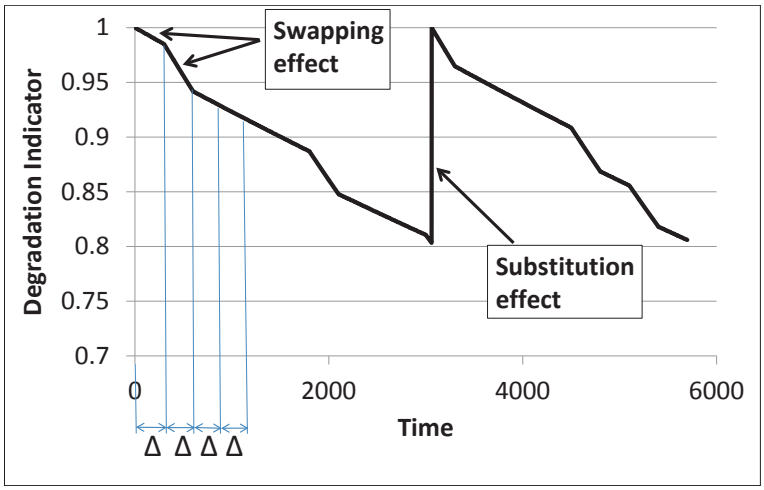


Figure 2.1: Health State Changes with Swapping and Substitution Actions.

In this formulation, the placement decision variable, $X_{ij}(k) \in \{0, 1\}$ in the model is chosen to be binary:

$$X_{ij}(k) = \begin{cases} 1 & \text{if the } i^{th} \text{ battery is placed in the } j^{th} \text{ loading profile at time } k \\ 0 & \text{if no action is taken at time } k \end{cases} \tag{2.1}$$

where its indices stand for

$$\begin{aligned} i = 1, \cdots, n \quad & \text{battery or vehicle index in the fleet} \\ j = 1, \cdots, m \quad & \text{degradation (loading) profiles} \\ k = 1, \cdots, K \quad & \text{discrete time, where } K \times \Delta = T = \text{plan horizon} \end{aligned}$$

For example, if $\Delta = 1$ month, and $X_{31}(7) = 1$ means that the 3^{rd} battery is placed in the first degradation profile at the 7^{th} month.

There are several constraints which are related directly to the placement decision variable. Some of these constraints arise from physical sense, others from demands and capabilities. The first constraint relates to the physical sense that a specific battery can be only assigned to one degradation profile for a specific interval. Additionally, the demand d_j drives the number of batteries (or vehicles) assigned to the jth degradation profile per interval. In formulation, these two constraints, respectively, translate to:

$$\sum_{j=1}^{m} X_{ij}(k) = 1 \qquad \forall i = 1, \cdots, n; \quad \forall k = 1, \cdots, K \tag{2.2}$$

$$\sum_{i=1}^{n} X_{ij}(k) = d_j \qquad \forall j = 1, \cdots, m; \quad \forall k = 1, \cdots, K \tag{2.3}$$

The placement variable is the indirect indicator for whether a swapping action has taken place or not. This can be formulated through:

$$|X_{ij}(k) - X_{ij}(k-1)| = \begin{cases} 1 & \text{if the } i^{th} \text{ battery is swapped at time } k \\ & \text{to/from the } j^{th} \text{ degradation profile} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The total number of swapping actions which takes place at time k can be given by:

$$Total\ number\ of\ swapping\ actions = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} |X_{ij}(k) - X_{ij}(k-1)| \tag{2.5}$$

Equation (2.5) enables us to formulate the constraints related to the company's preferential rules for swapping. Examples of these rules include an enforced minimum span between subsequent swapping actions for the same battery, and maximum number of allowable swapping actions within the fleet per interval. For the first one, if Δ is assumed to be equal to 1 month (for example), and a minimum of 3 months of enforced span between subsequent swapping actions for the same battery, then it translates to:

$$\frac{1}{2}\sum_{j=1}^{m}|X_{ij}(k)-X_{ij}(k-1)|+\frac{1}{2}\sum_{j=1}^{m}|X_{ij}(k+1)-X_{ij}(k)| \\ +\frac{1}{2}\sum_{j=1}^{m}|X_{ij}(k+2)-X_{ij}(k+1)|\leq 1,\quad \forall i=1,\cdots,n;\quad \forall k=2,\cdots,K \tag{2.6}$$

Or it can be abbreviated as:

$$\frac{1}{2}\sum_{h=0}^{2}\sum_{j=1}^{m}|X_{ij}(k+h)-X_{ij}(k+h-1)|\leq 1,\quad \forall i=1,\cdots,n;\quad \forall k=2,\cdots,K \tag{2.7}$$

In the general form, the constraint can be represented as (for a minimum span of $H\cdot\Delta$ between swapping actions for the same battery):

$$\frac{1}{2}\sum_{h=0}^{H-1}\sum_{j=1}^{m}|X_{ij}(k+h)-X_{ij}(k+h-1)|\leq 1,\quad \forall i=1,\cdots,n;\ \ \forall k=2,\cdots,K \tag{2.8}$$

Maximum number (α) of swapping actions per interval can be easily modeled as:

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}|X_{ij}(k)-X_{ij}(k-1)|\leq \alpha,\qquad \forall k=2,..,K \tag{2.9}$$

2.2.2 Substitution Decision Variables

A substitution decision variable, $Z_i(k) \in \{0, 1\}$ to represent any substitution action is included in the modeling.

$$Z_i(k) = \begin{cases} 1 & \text{if the } i^{th} \text{ battery is substituted at the beginning of epoch } k \\ 0 & \text{no substitution at the beginning of epoch } k \end{cases} \tag{2.10}$$

The substitution variable has only two indices as it relates only to battery i being substituted and time k at which substitution takes place.

The decision whether to initiate a substitution action or not, is merely dependent on the health state of the battery. This indicates the need to track the battery's health state degradation throughout its deployment in the field. In modeling DBOS with deterministic states, it is assumed that the degradable health states are predictable. The prediction is dependent on both the battery health state at the beginning of the current interval, and the degradation profile at which the battery is placed.

To track the degradation of the batteries health states, an accumulative degradation dependent quantity $y_i(k)$ is defined. The accumulative degradation is a monotonically increasing dependent variable which is calculated in the model based on the decision variables (placement and substitution variables). Based on the assumption of linear degradation the accumulative degradation can be found by:

$$
\begin{aligned}
y_i(0) &= 0, && \forall i = 1, \cdots, n \\
y_i(k) &= (1 - Z_i(k)) \left((y_i(k-1) + \sum_{j=1}^{m} r_j X_{ij}(k) \right) + Z_i(k) \sum_{j=1}^{m} r_j . X_{ij}(k) \\
&= (1 - Z_i(k)) y_i(k-1) + \sum_{j=1}^{m} r_j . X_{ij}(k) && \forall k = 2, .., K; \forall i = 1, .., n
\end{aligned}
\tag{2.11}
$$

where r_j is the degradation rate when the battery is assigned to jth degradation profile. In this formulation, when a new battery is brought in, the accumulative degradation is set to zero. Additional constraints arise from the bounds on the accumulative degradation variable:

$$
0 \leq y_i(k) \leq \beta, \qquad \forall k = 1, \cdots, K; \forall i = 1, \cdots, n \tag{2.12}
$$

where β is the threshold at which substitution becomes inevitable.

2.2.3 Objective Functions

There are several objectives that could be used towards an optimum policy. The policy can aim for minimized maintenance costs, maximized utilization, or a combination of both. One direct and simplified objective that can be chosen is to minimize the projected maintenance costs over a finite plan horizon. With the satisfaction of the constraints described above, the minimization of the projected costs which are attributed to the substitution and swapping actions can achieve an optimum scheduling policy. Based on the discussion previously, the cost can be found by:

$$
J = \frac{1}{2} \sum_{k=2}^{K} \left\{ c_1(k) \sum_{i=1}^{n} \sum_{j=1}^{m} |X_{ij}(k) - X_{ij}(k-1)| \right\} + \sum_{i=1}^{n} \sum_{k=1}^{K} \{ c_2(k) Z_i(k) \} \tag{2.13}
$$

where $c_1(k)$ is time dependent swapping cost coefficient, which includes penalties and potential of loss due to swapping, and $c_2(k)$ is time dependent substitution cost coefficient. The choice to make both cost coefficients as time dependent increases the flexibility of the model.

2.3 On the Solution of Deterministic DBOS Model

The mathematical model of the DBOS policy with deterministic states has been introduced in Section 2.2. This section is dedicated to the solution of the generated model.

2.3.1 Preliminary Attempts and Limited Success of Standard Stochastic Optimization Algorithms

Although the generated model successfully captures the intended functionality of the policy, the DBOS policy model is a Zero-One Integer Nonlinear programming (ZOINLP) problem, whose solution will be shown to be challenging. More specifically, all the decision variables (both placement and substitution variables) are zero-one integers and the only non-linearity in the model arises from the accumulative degradation update Equation (2.11). The outer structure of the model suggests simplicity and the potential of a simple algorithm application to find the optimal solution. However, the model's "looks are deceiving" as the inter-relations between the decision variables in addition to the significantly larger coefficient of the substitution cost with respect to the swapping cost inflicts complications through the application of different algorithms. The complexity in the inter-relations of the decision variables lies on the fact that there is no direct effect between them, rather placement variables influence the calculation of an intermediate quantity (the accumulative degradation) which influences the other decision variables (substitution variables). The coefficient of the substitution cost is estimated to be of two orders more than the coefficient as-

signed to the swapping cost. In fact this corresponds to the core motivation in DBOS policy as the substitution is considered extremely costly with respect to swapping.

The deterministic nature of the model suggests an attempt to find the optimal solution through a deterministic algorithm. We define deterministic algorithm to be the one which generates the same answer for different runs. Exhaustive Search is one of the most known deterministic algorithms which is used usually as a bench mark due to its ability in finding the global optima. However even for a small DBOS problem, the exhaustive search is unattainable. For example, the problem in Case Study I in Section 2.4.1 is expected to have an execution in the order of ($2^{64} = O(10^{19})$) function evaluations. This is an astronomical number.

The B&B algorithm becomes deterministic if the applied sub-solver for the relaxed problem is deterministic as well. An example of this is B&B with Simplex for linear problems. The problem on hand is not linear which means that the sub-solver should be built based on a NLP algorithm. SQP with local search is not purely deterministic; however it is considered to be the most efficient general purpose NLP algorithm today [72]. In fact, the modified version of SQP approaches deterministic behavior when the algorithm converges. Motivated by that, a B&B algorithm with SQP and local search has been attempted. The optimizer failed to arrive at feasible answer. The reason behind this failure is the extreme nonlinearity growth in the accumulative degradation constraint (Equation (2.11)). The growth in the nonlinearity of this constraint has been shown explicitly in Appendix A. Preliminary findings showed that the sub-solver is greatly struggling in solving the relaxed problems and thus the algorithm is incapable of finishing one node of the enumerated tree. There are some modifications in the implementation of the B&B algorithm which could help the algorithm to overcome this highly nonlinear constraint, or even make it immune by choosing a more adequate sub-solver. Later, we use this conclusion in the development of the DBOS-specific B&B-based optimization algorithm in Section 2.3.2.

The failure in applying B&B algorithm with SQP has motivated the consideration of stochastic algorithms (Heuristics). Additionally, standard stochastic algorithms such as GA and SA do not apply intensive calculations per iteration, making them increasingly attractive for problems with large number of variables. As GA and SA deal only with unconstrained optimization, the problem was relaxed using Lagrangean multipliers.

As we will show in Section 2.4, both GA and SA were applied. The applied GA has a built-in elitist strategy, where the highest-ranking solution of all the solutions produced by the previous generations is copied directly into the next generation. Applying this helps in retaining most important genes within the population pool, and the best objective value in each generation is assured not to increase throughout all the iterations. SA, on the other hand, was applied with a standard form. The neighborhood function which generates the new solution is chosen to be of random nature. The neighborhood function works extremely well when the initial solution the algorithm starts with has all zeros at the placement variables. This means that the penalized function undergoes extreme descending in the first iterations until the heavily penalized equality constraints (Equations (2.2) and (2.3)) are satisfied. After that the convergence slows down. Based on the general form of SA, solutions that do not improve the objective function are only accepted if the acceptance criteria allow for that. This criteria depends on the random number generated, the difference between the objective values in the current and previous iterations, and the current iteration (temperature). Thus with adequate cooling temperature schedule, diversity in the new solutions will resume even after equality constraints are satisfied.

As it will be shown in the next Section, both GA and SA have been able to generate relatively good solutions for small problems. While most of the generated solutions outperformed the No Swapping policy, the majority of the solutions were suboptimal. This is due to the fact that both algorithms were converging at local

optima. We note that the SA implementation outperformed the GA implementation. This was observed in the speed of convergence and the quality of the output results. However, using the DBOS-Discretization interval correspondence detailed in Section 2.2.1, both algorithms have been found to fail the test. This reflect on the robustness of the both algorithms in terms of finding solution for problems with large number of decision variables. In addition to lack of robustness, repeatability (because of the stochastic nature of heuristics by construction) was another issue. As we are generating solutions for the first time for a new problem, we are lacking information of how good a solution is. For these reasons a retrofitting of the B&B algorithm that applies specifically to DBOS has been implemented.

2.3.2 DBOS-Policy-Specific Branch-and-Bound-based Optimization Algorithm

2.3.2.1 Proposed Algorithm Explanation

In this Section, we introduce a DBOS-policy-specific Branch-and-Bound-based algorithm that will be shown to successfully generate repeatable answers as well as expand the scalability towards DBOS problems of larger sizes. The algorithm is illustrated in Figure 2.2.

The algorithm reduces the complexity of the model by providing incremented estimates of the total number of required substitutions ($\sum_{i=1}^{n}\sum_{k=1}^{K} Z_i(k)$). The estimates are generated heuristically from expected loads and logic induced rules (rules that are sensible in the manner of fleet management and expected substitution occurrences). Total demand over horizon (when averaged per battery) dictates whether this estimate is started at zero or not. For example, if the average demand per battery exceeds threshold value (β), starting with estimate ($\sum_{i=1}^{n}\sum_{k=1}^{K} Z_i(k)$=0) becomes trivial. For each estimate, all satisfying (non-repeated) configurations are investigated. The reconfiguration is done systematically that it will generate each time a

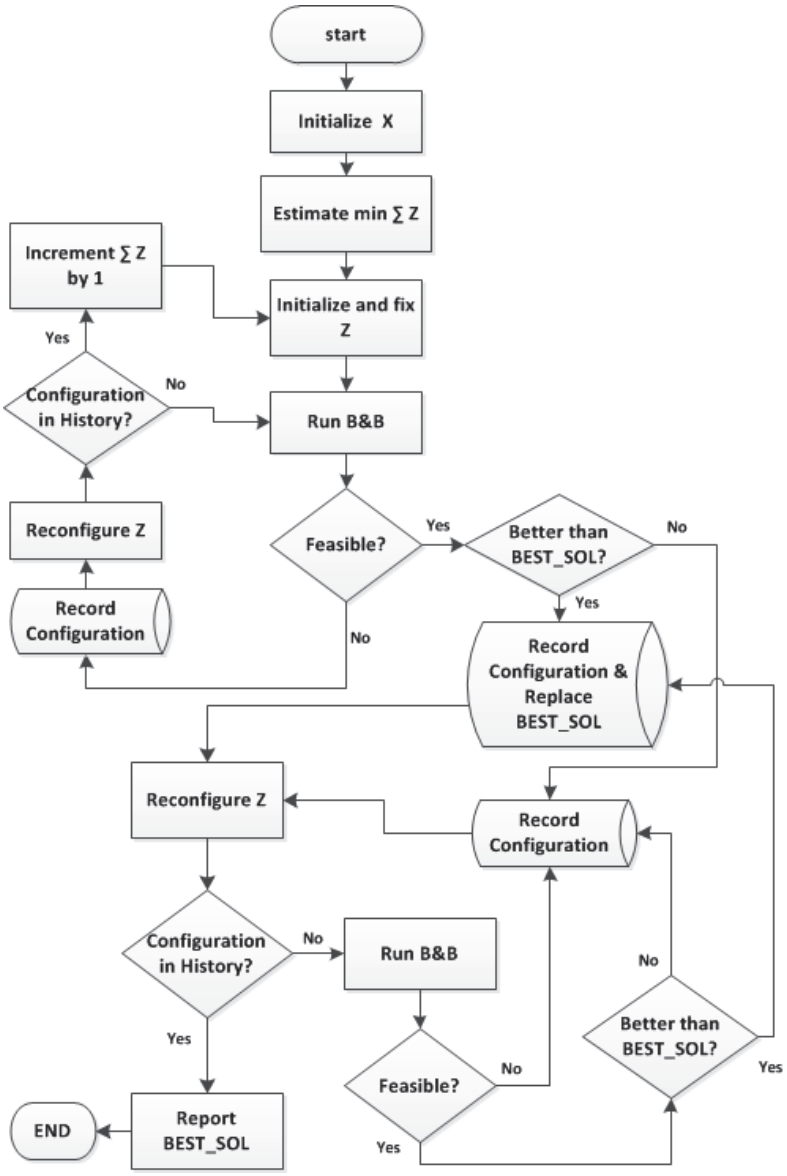


Figure 2.2: DBOS-Policy-Specific Branch-and-Bound-based Algorithm.

new configuration until all possible unrepeated configurations for that estimate have been tested. We note here that repeated configurations include any new $Z_i(k)$ array that is generated from swapping rows in an old $Z_i(k)$ array as this action provides no new configurations. The first estimates are chosen to be very conservative (low number of substitutions). This probably leads to infeasibility for all or most reconfigurations of $Z_i(k)$ for the first iteration. Nevertheless, the conservativeness provides

assurance for minimum objective value function as the major part of the cost is attributed to the substitution. We note here that the infeasibility is identified quickly and therefore the performance of the algorithm in general is not hindered by the conservativeness.

With this implementation, at each instant the nonlinearity in the model (Equation (2.11)) ceases to exist and the problem is reduced to a Zero-One Integer Linear Programming (ZOILP) problem. This promotes the utilization of a B&B scheme with a (LP) sub-solver. The later only applies if the absolute value in the objective function is formatted in the standard LP form as well. This is implemented through a number of well-known mathematical tricks that do not solve a 100% equivalent problem; rather they solve another problem that has solution of objective value similar to the original problem. Section 1.3 in Bertsimas and Tsitsiklis's LP book [16] has a detailed explanation of the absolute value handling using either one of two tricks (See Figure 2.3). The first relies on introducing new variables replacing all the absolute values. Additional constraints then make certain that whatever was inside the absolute value in the original problem is less than these new variables. The second trick also depends on introducing new variables. These variables are constrained to be nonnegative and they replace the original problem variables in both the objective function and the constraints. The second trick is obviously less attractive if the absolute value is an interaction of more than one variable. It should be noted that the formatting of the absolute value into the standard LP form with these tricks incorporates an increase in the number of the decision variables which may adversely affect the algorithm's performance for extremely large problems.

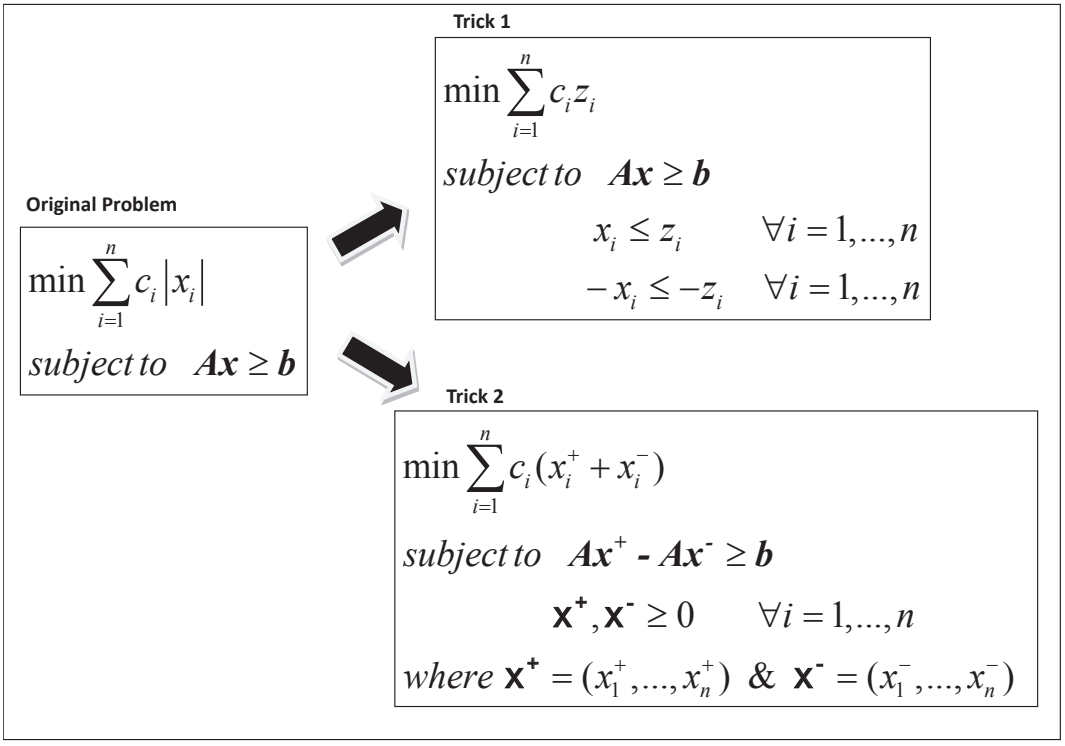


Figure 2.3: Handling Absolute Values in LP Problems [16].

2.3.2.2 On the Optimality of the Generated Solutions by the Proposed Algorithm

The proposed algorithm solutions can be characterized to be global optimal solutions. This can be proven through investigating the nature of the three levels of optimization over which the algorithm is operating. The lowest level represented by solving the relaxed reformulated (to handle the absolute value) LP problem is solved by revised simplex method with anti-cycling, which finds global optimal solutions [16] drawn from the convexity of the objective function. The second level is the branching search to find the zero-one solution that minimizes the objective function in the branch and bound. As the branch and bound algorithm searches all possible branches (except when extending the branches is proven not to enhance the objective value), the branch and bound finds a global optimum for its problem. Finally, the third and highest level is the search for the configuration that satisfies a specific estimate of

($\sum_{i=1}^{n}\sum_{k=1}^{K} Z_i(k)$) (determined by the first loop in the algorithm), and has the best objective value. With the satisfaction of the condition which states that the substitution is much more costly than swapping (one of the conditions stated in Section 1.2 for utilization to be promotable by DBOS), the solution found by the third level is a global optimal one.

2.3.2.3 Limitations of the Proposed Algorithm

Despite its ability to outperform GA and SA (as it will be shown in the next section), this algorithm has limitations as well. First, the prescribed procedure in the algorithm is aimed at DBOS compatible problems described in the beginning of Section 1.2. Mainly the substitution cost must be significantly larger. The algorithm utilizes the fact that any number of swapping actions will always be cheaper than additional substitution action. If this does not apply, a retrofit of the procedure should be included, where the second loop is no longer running configurations of one estimation of ($\sum_{i=1}^{n}\sum_{k=1}^{K} Z_i(k)$), but rather several ones. This procedure will be used in Section 2.4.2 to obtain solutions for the parametric studies there. It will be shown then that the DBOS model still applies in these instances, but rather they are no longer candidates of utilization promotion achieved by optimal swapping.

Another major limitation to the proposed algorithm is the effect of the problem size when the problem is getting significantly large. As we mentioned, the algorithm will be shown to solve larger instances than GA and SA can, yet when the instance size increases significantly, the algorithm can become computationally demanding. When the problem size increases, the computational time increases in three directions according to the algorithm procedure. The first is the number of configurations satisfying a specific estimate of ($\sum_{i=1}^{n}\sum_{k=1}^{K} Z_i(k)$) found from the first loop. The second is the number of nodes (branches) potentially examined in the branch and bound algorithm's search to find the solution. The third is the size of the LP relaxed

problem solved at each node by Simplex method, which is significantly affected by the problem size (specifically, the number of decision variables). Therefore, it can be easily concluded that the algorithm will demand significant computational time when the problem size is significantly large.

One way to reduce this effect, is parallelism, where the application of the second loop (the solution of the different configurations) is conducted on parallel solvers (or machines). Furthermore, the previous procedure can be enhanced if the parallel solvers are sharing the best solution found, where this can be used as an upper bound, to further truncate the branch and bound "search tree". This means that if some solver is getting the solution of the relaxed LP at a specific branch to be higher than the shared best solution, the whole branch can be cut and ignored. This can significantly reduce the total computational time, as we are no longer solving many configurations (in the second loop) completely due to this truncating effect.

2.4 Case Studies

In this section, we report numerical results of the different optimization algorithms discussed above on the formulated DBOS model.

2.4.1 Case Study I

The problem parameters are available in Table 2.1. Initially, the cost coefficients and degradation rates are inspired by real applications. However, for the sample problem to be presentable and comprehensible, the maintenance plan horizon is shortened. Therefore, the degradation rates have been modified to reflect this. The modification in the coefficients is intended to simulate the real scenario where longer horizons are chosen, and thus substitutions are inevitable.

Table 2.1: Case Study I Parameters

Parameter	Symbol	Value
Number of vehicles in the fleet	n	5 (A,B,C,D,E)
Number of Loading Profiles	m	4
Plan Horizon (years)	K	4
Discrete interval (years)	Δ	1
Demand	d_j	[1, 1, 1, 2]
Degradation Rates	r_j	[0.11, 0.08, 0.04, 0.02]
Swapping Cost Coefficient	$c_1(k)$	\$400
Substitution Cost Coefficient	$c_2(k)$	\$11600
Threshold	β	0.2

2.4.1.1 Results from SA, GA, and the Proposed Algorithm

GA is first applied. Figure 2.4 shows a convergence for one of the runs. Elapsed time for each run was about 19 minutes, when run on a 2.67 GHz quad-core processor.

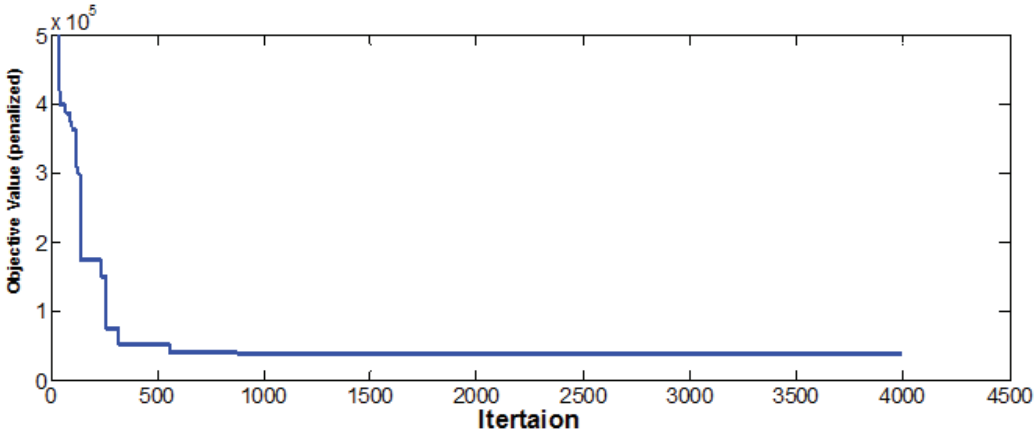


Figure 2.4: Convergence in GA When Applied to DBOS Model.

The corresponding solution associated with this run has an objective value of $\$26,400$ where all constraints are satisfied. The solution found by the algorithm is shown where $X^*_{ij}(k)$ represents the placement variables, $Z^*_i(k)$ represents the substitution variables, and $y^*_i(k)$ corresponds to the accumulative degradation.

The results above can be summarized in the following schedule of batteries place-

$$k \qquad 1 \qquad\qquad 2 \qquad\qquad 3 \qquad\qquad 4$$

$$X^*_{ij}(k) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Z^*_i(k) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad y^*_i(k) = \begin{bmatrix} 0.02 & 0.04 & 0.06 & 0.17 \\ 0.04 & 0.08 & 0.12 & 0.20 \\ 0.11 & 0.11 & 0.13 & 0.15 \\ 0.08 & 0.10 & 0.11 & 0.15 \\ 0.12 & 0.10 & 0.18 & 0.20 \end{bmatrix}$$

ment, where the numbers refer to the degradation profile at which the battery is placed at each interval:

At the end of the first year, the optimizer swaps batteries D and E (D↔E), then at the end of the second year it swaps batteries C and D (C↔D), in addition to performing a substitution on battery C prior to that. At the end of the third year, the optimizer swaps batteries A, B, D, and E. Battery A takes the place of battery D (A→D), battery D takes the place of battery B (D→B), battery B takes the place of battery E (B→E), and finally battery E takes the place of battery A (E→A). A replacement for battery D takes place as well at that year.

It is clear how the model is able to capture all the intended swapping actions to minimize the number of substitutions. In fact, when compared with no swapping policy (direct policy) which results in a cost of $46,400, the DBOS policy model savings in this run amount to 43.1% of the projected costs. A summary of several

Table 2.2: Schedule of Batteries Placement from One of GA Results

Battery	1st year	2nd year	3rd year	4th year
A	4	4	4	1
B	3	3	3	2
C	1	1*	4	4
D	2	4	1*	3
E	4	2	2	4
(*) means a substitution action has taken place				

GA runs is shown in Figure 2.5.

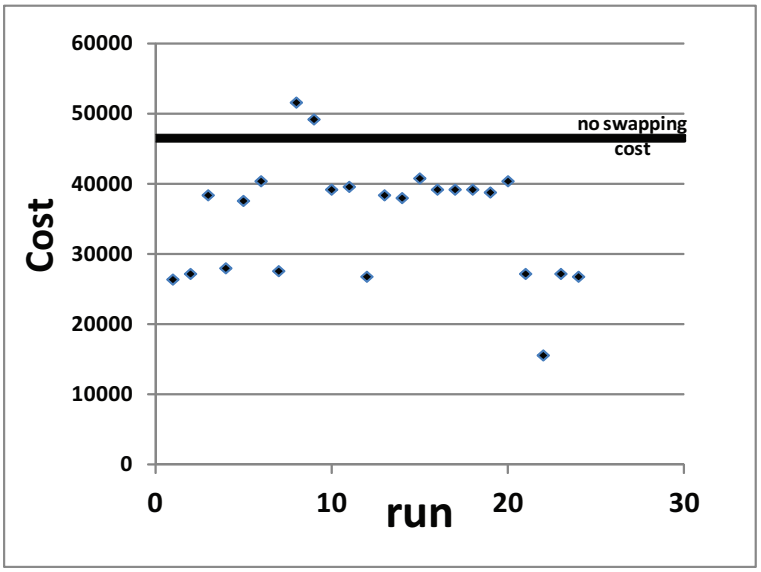


Figure 2.5: GA Different Runs.

SA algorithm is implemented next. Figure 2.6 shows 30 different runs for the SA implementation. From the figure, it can be seen that the SA implementation for the DBOS model outperformed the GA implementation. The quality of the solutions is enhanced as all results outperformed the No Swapping (direct) policy and the solutions have lower cost values in general. In addition to the quality of the output results, the speed of convergence was better in SA as well. Elapsed time for each run was only 45.7 seconds.

We apply the DBOS-specific B&B-based optimization algorithm. The results are shown in Figure 2.7. It can be clearly seen how the proposed algorithm successfully and repeatedly achieve the global optimal solution of cost equivalent to $14,000; a solution that has not been attained with either GA or SA. The optimum schedule per DBOS policy for Case Study I is shown in Table 2.3. From the table, it can be seen that there was a way with strategically selected swapping actions to suffice with one substitution. While one substitution results were achieved by two runs from each of GA and SA results (the lowest occurrences in Figures 2.4 and 2.6), the proposed

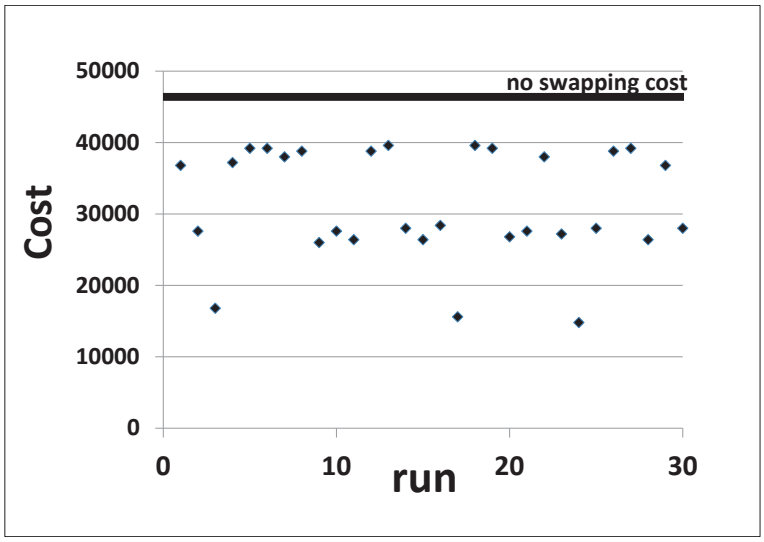


Figure 2.6: SA Different Runs.

algorithm outperformed them in reducing the number of swapping actions to arrive at one substition result.

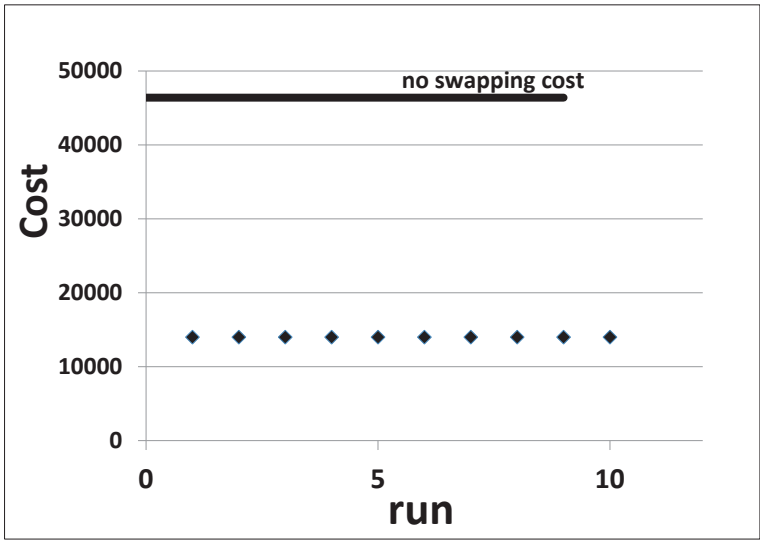


Figure 2.7: Proposed Algorithm Different Runs.

We finally present the results of the DBOS-discretization interval correspondence test for Case Study I when SA and the proposed algorithms were used (GA results were significantly worse and hence ignored). Δ was decreased to half a year and then

Table 2.3: Schedule of Batteries Placement from the Proposed Algorithm Results

Battery	1st year	2nd year	3rd year	4th year
A	4	4	4	1
B	2	1	1*	2
C	4	2	2	4
D	3	3	3	3
E	1	4	4	4
(*) means a substitution action has taken place				

to quarter a year. Figure 2.8 summarizes the results. It can be seen how SA is failing when the problem size is getting larger, while the proposed algorithm maintains an equal cost. A clearer view of the DBOS-discretization interval correspondence is found in Case Study II (Section 2.4.2).

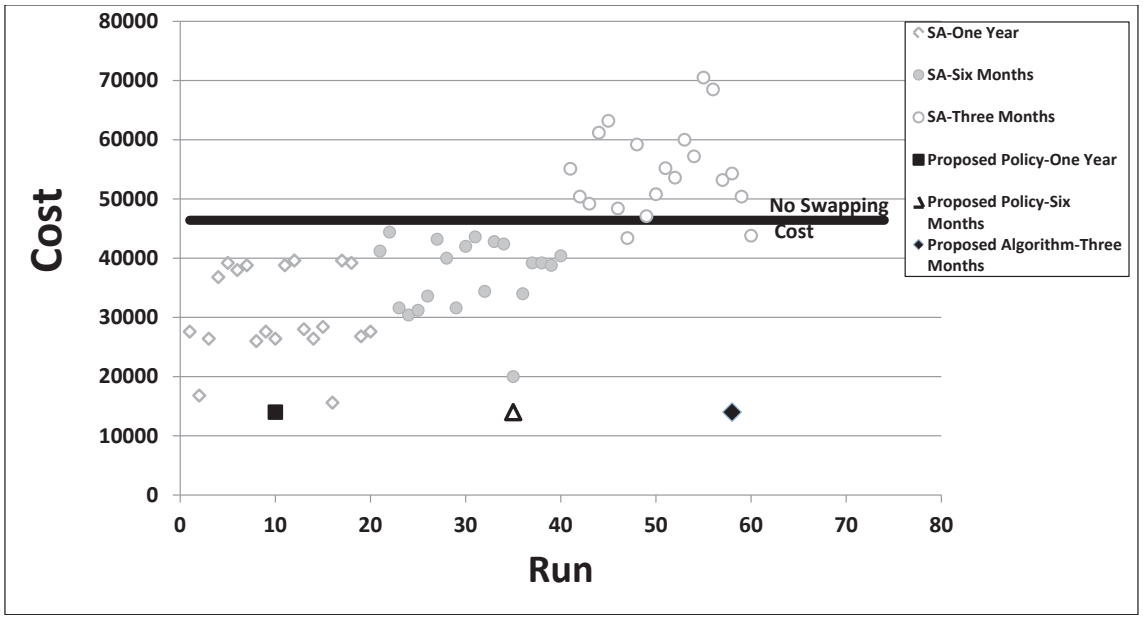


Figure 2.8: SA Algorithm Results vs. DBOS-Specific B&B-based Algorithm Results When Δ is Reduced in Case Study I.

2.4.1.2 Benchmarking DBOS Policy

Finally for Case Study I, we benchmark the performance of the DBOS policy, several management policies have been applied (see Figure 2.9). The maintenance

plan cost has been evaluated for each of the four shown policies. In the "No-swapping" policy, the batteries in the fleet are dedicated to one degradation profile throughout the plan horizon, where no swapping is allowed. The rotational fixed swapping policy refers to the policy where swapping actions are conducted in a timely, fixed and cyclic manner. An example of that is the rotational swapping of tires in automobiles to even out the degradation. The third policy (Intelligent fixed swapping) refers to the case when swapping actions are conducted between the most and the least degraded batteries at each cycle. The intelligence refers to basing the decision on being informed about the health state of the battery. Though the latter performs better than the No-swapping and Rotational Fixed Swapping policies, the DBOS policy clearly outperforms all of them. Moreover, the DBOS policy coupled with the proposed optimization algorithm represents the best result with the least maintenance plan projected costs.

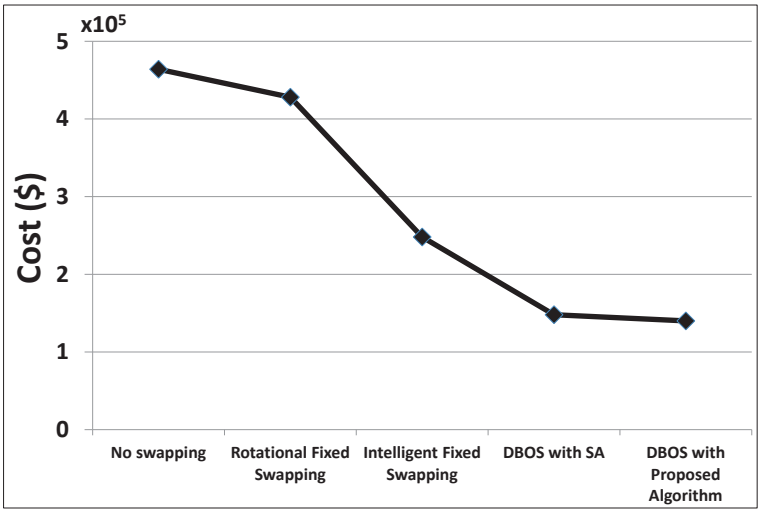


Figure 2.9: Benchmarking DBOS Policy.

2.4.2 Case Study II

In this section, we perform several parametric studies. We first verify the DBOS-discretization interval correspondence we concluded in Section 2.2.1. Then, we per-

Table 2.4: Case Study II Parameters

Parameter	Symbol	Value
Number of vehicles in the fleet	n	3
Number of Loading Profiles	m	2
Plan Horizon (years)	K	4
Discrete interval (years)	Δ	1, 2/3, 1/2, 1/4
Demand	d_j	[1, 2]
Degradation Rates	r_j	[0.11, 0.04]
Swapping Cost Coefficient	$c_1(k)$	\$400
Substitution Cost Coefficient	$c_2(k)$	\$11600
Threshold	β	0.2

form parametric studies for the cost coefficients and the degradation rates to confirm the rational of DBOS illustrated in Chapter I.

For the first part, the DBOS-specific branch-and-bound-based algorithm is implemented on the case study of parameters shown in Table 2.4, where Δ is varied from 3 months to 6 months to 8 months and finally to 1 year. The case study will serve as well to illustrate the scalability of the proposed algorithm when larger numbers of decision variables are involved. That is, decreasing Δ increases the size of the problem significantly due to the increase in the placement and substitution variables under investigation. The influence of this increase on the performance of SA and the proposed algorithm is investigated.

The left-hand plot in Figure 2.10 shows the results when SA was used. It can be seen that the SA algorithm is unable to capture the intended behavior of the DBOS policy. The policy aims to opt for swapping when swapping achieves decreased objective values. In this case, as the problem size grows the optimizer fails to recognize the unnecessary swapping actions and therefore the total cost increases. On the other hand, the right-hand plot in Figure 2.10 shows the results of the DBOS-specific B&B-based algorithm when Δ is varied. The anticipated behavior appears clearly. The cost decreases when Δ is varied from 1 year, to 8 months and finally to 6 months. After that, there is no improvement in the objective value when Δ is shortened from

6 months to 3 months. The optimizer in this case opts for no more swapping actions than what has been chosen for the 6 months discretization interval, and therefore the policy is correctly captured.

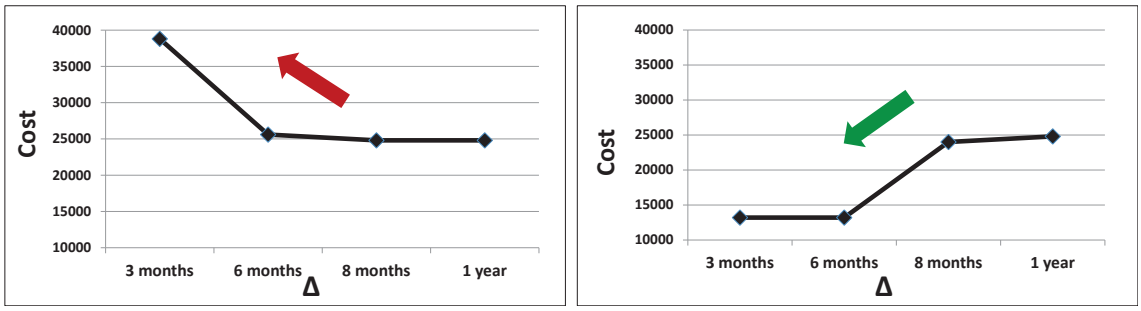


Figure 2.10: SA Best Result (Left) and DBOS-specific B&B-based Algorithm Result (Right) When Δ is Varied.

For the second part of this case study, we variate the cost coefficients associated with the substitution and swapping actions bringing them closer to each other. Per the discussion in Section 1.2, the DBOS policy will enable further utilization through conducting optimal swapping decisions when the substitution cost is significantly larger than the swapping cost in addition to further conditions. We demonstrate that when we bring the cost coefficients closer to each other (value-wise), the swapping will no longer be justified. Table 2.5 shows the results. We note first that all the results have been obtained by retrofitting the proposed algorithm as described in Section 2.3.2.3. We also make note that we are not concerned by the optimal values themselves (as these are numeric values depending on the chosen cost coefficients). We are rather more interested in the number of substitutions and swapping actions, of which the optimal solution is comprised. The results in the table display the fact that when the swapping cost becomes close to the substitution cost (for this small case study it is exactly equivalent), swapping-promoted utilization will be no longer attainable as substitution provides cheaper (and better for the "future") results.

In the final part of this case study, we variate the degradation rates maintaining

Table 2.5: Parametric Study of Swapping and Substitution Costs

$c_1(k)$	$c_2(k)$	Optimal Cost	# of Subs.	#of Swaps.	Enhancement Over No Swapping
800*	11600*	24800	2	2	47%
2500	7500	20000	2	2	50%
5000	5000	15000	3	0	0%
(*) The Original Values in the Case Study					

similar total loads per fleet. In Section 1.2, we stated that the swapping will promote utilization if the degradation rates are sparse (scattered) enough to make the swapping a "meaningful" action. This means that if the degradation profiles are close to each other, the swapping will not promote utilization and hence the DBOS policy will opt for more substitutions over swapping actions. The results confirming this are shown in Table 2.6. Once again, the results are obtained by the retrofit of the DBOS-specific B&B-based algorithm to account for the cases when substitution is preferred over swapping actions. Also we are not concerned with the optimal costs in terms of values. We are rather concerned with the configuration of substitutions and swapping actions that led to it. It can be seen that once the degradation rates are close to each other (not sparse enough), the swapping no longer provides utilization promotion, and hence is no longer optimal.

Table 2.6: Parametric Study of Degradation Rates

Degradation Rates	Optimal Cost	# of Subs.	# of Swaps.	Enhancement Over No Swapping
$[0.13, 0.02, 0.02]$	24800	2	2	47%
$[0.11, 0.04, 0.04]^*$	24800	2	2	47%
$[0.08, 0.06, 0.05]$	23200	2	0	0%
$[0.07, 0.06, 0.06]$	34800	3	0	0%
(*) The Original Values in the Case Study				

2.5 Conclusions

This chapter has presented the formulation of degradation-informed resource allocation policy denoted as Deterministic Degradation-based Optimal Swapping (DBOS). The formulation was based on the primary application in this thesis which is the fleet level battery utilization. The policy which is intended to be part of the maintenance planning for the fleet, utilizes batteries on fleet level through a series of optimally chosen swapping and substitution actions. The policy takes advantage of the different degradation rates of the batteries within the fleet, based on loading conditions, to choose optimal placements of these batteries. The representative mathematical model with deterministic health states have captured the intended functionality of the policy through correct optimization. Investigation of standard optimization algorithms performance with DBOS has been presented as well. A DBOS-policy-specific algorithm has been developed and successfully implemented, where numerical results have shown the outstanding performance of the algorithm in comparison to other standard optimization techniques. Numerical results, as well, validated the role of the discretization interval in the DBOS policy denoted as the DBOS-discretization interval correspondence. Upon achieving global optimal results, decreasing Δ is shown to allow the option to (but not necessary) perform additional swapping actions minimizing the costly substitution ones. Finally DBOS policy has been benchmarked with other maintenance management policies where the coupling of DBOS with the proposed optimization algorithm has provided significant savings in the projected maintenance costs.

CHAPTER III

Stochastic Degradation-based Optimal Swapping

3.1 Introduction

In the previous chapter, a high confidence (deterministic) degradation estimation has been adopted to uniquely formulate the degradation-informed resource allocation planning scheme, designated as Degradation-based Optimal Swapping (DBOS). In this chapter, the problem is extended to include uncertain degradation estimation and the formulation of the Stochastic DBOS policy. The latter will mainly depend on the adoption of Markov decision processes (MDP) framework. By concept, policies are more adaptive than planning and scheduling schemes. Policies react to "bad luck" occurrences with optimal decisions, preventing undesirable outcomes that may result from inaccurate prediction. For example, batteries can degrade more than the anticipated values at a specific interval, and thus the schedule no longer represents an optimal management scheme. Therefore, to ensure the achievement of the main goals of the DBOS concept, uncertainty has to be introduced. The research in this chapter includes the necessary reconfigurations of the deterministic DBOS model to include the uncertainty and the challenges in fitting it to the (MDP) framework.

Literature reveals numerous approaches for planning with uncertainty. These can be categorized into six groups: two-stage stochastic programming, parametric programming, fuzzy programming, chance constraint programming, robust optimization

techniques, and risk mitigation approaches [73]. The differences in these main approaches lie in the way uncertainty is incorporated and dealt with in the model. There are three different ways for uncertainty incorporation. The bounded distribution form, where the uncertain parameter is assumed to take on values within a specified range defined by an upper and lower bound, is often adopted if there is not enough information to construct an accurate estimation of the uncertain parameter distribution [73]. The use of probabilistic distribution can take place when there is sufficient information to construct a reliable distribution for the uncertain parameter. Finally the use of fuzzy sets as noted by Li and Ierapetritou [74] can replace the bounded and known distribution cases.

Parametric Programming, which is based upon the theory of sensitivity analysis, aims to define a function which maps the uncertain parameter values to a given optimal solution for the entire uncertain parameter space. Examples of this work are found in [75, 76, 77]. Chance constraint programming replaces constraints containing uncertain parameters with their probabilistic forms. Using the distributions, the probabilistic constraints are then reformulated into a deterministic form. The major issue of chance constraint programming is that feasibility of the solution is not guaranteed. Examples of chance constraint programming can be found in [78, 53]. Unlike chance constraint programming, robust optimization techniques guarantee the feasibility of the obtained solution for the nominal set of system conditions, as well as being robust with regard to the multiple forms of uncertainty present within the system under investigation. The robust optimization framework involves first expressing the true parameter values through the declaration of random variables, then the formulation of probabilistic constraints, and finally, the transformation of these probabilistic constraints into their deterministic counterparts, which are added to the existing model [73]. Examples of robust optimization techniques can be found in [79, 80].

Two-stage stochastic programming, which is the primary approach in transportation planning, represents the umbrella under which several well-known techniques such as Multi-stage Stochastic Programming and Stochastic Dynamic Programming (SDP) lie. The later is sometimes referred to in some publications as Markov Decision Processes (MDP). The Two-stage programming is set based on the notion of a first-stage that contains variables which must be ascertained before the uncertain parameters are realized, and a second stage that is composed of those variables which represent recourse decisions that are enacted upon the realization of the given uncertain parameters [73]. The first-stage objective function term is deterministic in nature while the second-stage term involves an expectation evaluation. In order to address the expectation evaluation, a finite number of uncertain parameter scenarios can be generated with the recourse variables being indexed by scenario so that every possible parameter realization has an associated recourse action. The scenario generation approach becomes problematic with the growth in the model size due to the increase in the number of considered scenarios.

Distribution-based approach can represent an alternative to the scenario generation, where the expectation of the recourse objective function term is determined through the integration of the given probability distributions. While the problem size is considerably smaller in the distribution-based approach, the formulation becomes nonlinear which requires techniques such as Monte Carlo, Gaussian quadrature, etc. Once again, these methodologies may become computationally expensive as the number of uncertain parameters increase.

Markov Decision Processes (MDP) represents the most famous sequential decision modeling technique. In it, the set of available actions, the rewards, and the transition probabilities depend only on the current state and action and not on states occupied and actions chosen in the past. The model is sufficiently broad to allow modeling most realistic sequential decision-making problems [15]. A close problem to DBOS

mentioned in the well cited reference is the bus engine replacement problem which is also known as Zurcher's replacement problem [14]. Further explanation of the problem and explanation of the similarity and differences with DBOS is available in Section 1.2.1.

3.2 Mathematical Modeling of Stochastic DBOS Policy

The transformation of the deterministic DBOS model to become a stochastic one relies on including uncertainties within the health state prediction. In more details, the health states of the batteries at the end of the next discrete interval in the time horizon are estimated with some uncertainty at the beginning of this interval depending on the decisions taken (placement and substitution). Once the interval is over, the real health state information becomes available and can be used to make decisions for the next interval. In current practice, there are several methods to achieve that. There could be offline testing taking place at the end of each interval, or an online data acquisition system which collects and analyzes data at the end of the interval. The details of acquiring the real health state information of the battery in this case is beyond the scope of this thesis.

The framework of the decision making will be based on Stochastic Dynamic Programming (SDP) and Markov Decision Processes (MDP). We first reformulate the original deterministic model to account for uncertainties through the following modification to Equation (2.11):

$$y_i(k) = (1 - Z_i(k))y_i(k-1) + \sum_{j=1}^{m} r_j X_{ij}(k) + \epsilon_i \quad \forall k = 2, \cdots, K; \forall i = 1, \cdots, n \quad (3.1)$$

Where ϵ_i represents the error in the deterministic degradation prediction, identified after the discrete interval has passed. At the start of the interval, both groups

of decisions; the placements $X_{ij}(k)$ and the substitutions $Z_i(k)$; are chosen. The uncertainty associated with the estimate will be in the form of a random variable ϵ_i that represents the error in the prediction. While this random variable can take several values (negative and positive), the probability distribution associated with it is assumed to be known. With this assumption, the stochastic dynamic programming (SDP) problem as in (Puterman, 1994) can be defined as:

$$\text{SDP} = \{\text{S, M, R, C, P}\}. \tag{3.2}$$

where S denotes the set of states, M denotes the set of available actions, R is the set of state dependent rewards, C is the set of state and action dependent costs, and P is the set of transition probabilities which depend on what action can result in what state(s) when executed from any of the states.

3.2.1 States

The states in SDP are a sufficient and efficient summary of the available information which affects the future of the stochastic process. Other than the partially observed MDP framework, the state at a point in time should not contain information that is not available to the decision maker at that time, because the decision is based on the state at that point in time. For this problem, the states must specify the health state of each battery (using the accumulative degradation) and the current "shape" of the system in terms of battery placements, which we will refer to as system placement (G). After decisions (actions) are made, the state of the system will change accordingly including degradation in the health states, and a change in the battery placements if a swapping action has been selected. Before the end of this current interval, the new health states can only be predicted with some uncertainty. Therefore, only the health states at the start of the interval are considered part of the

state definition in this SDP framework. For a concise formulation, the placements of all the fleet batteries in the loading profiles at a specific instant, represented by the binary placement variables ($X_{ij}(k)$), will be replaced by one discrete variable; called system placement (G); that summarizes all of them. There will be always a limited number (n_G) of possible (feasible) system placements as can be concluded from Equations (2.2) and (2.3). These equations limit the feasible combinations of $X_{ij}(k)$. For example, a small fleet comprising of 3 vehicles distributed on 2 loading profiles can only have 3 different placements. There is no single formula which can calculate n_G, but a very simple recursive program can find the number and all possible configurations.

The health states of each battery are once again tracked through the accumulative degradation. Y_{ip} represents the accumulative degradation of the ith battery in state ς_p. Due to the limited battery's operational range in terms of health state, Y_{ip} possible values are limited by the threshold value (at which substitution is inevitable), and the discretization resolution ϕ. For example, we will choose for our case studies the threshold value (β) to be 20% or 0.2. This is inspired by the definition of the End-of-Life for hybrid vehicles batteries occurring when the battery meets specific failure criteria (See Section 1.2.1). Therefore and as the accumulative degradation has been modeled to be monotonically increasing, the threshold of 20% is chosen. The discretization resolution (ϕ) will mainly depend on the accuracy of the methodology (offline testing, analysis of online data, etc.) through which the real health state is identified at the end of each discrete interval. The total number of states (N) in the state space can then be found by $(\left(\frac{\beta}{\phi}\right) + 1)^n \times n_G$.

$$S = \{\varsigma_1, \cdots, \varsigma_N\} \tag{3.3}$$

$$\varsigma_p = (Y_{1p}, \cdots, Y_{np}, G_p) \qquad p = 1, \cdots, n \tag{3.4}$$

$$Y_{ip} \in \{0, \phi, 2\phi, \cdots, \beta\}$$

$$G_p \in \{1, \cdots, n_G\}$$

3.2.2 Actions

For this problem, the actions (decisions) available to the decision maker are all the possible combinations of swapping-substitution actions. The decision maker can opt for a change in the system placement (G'), and hence swapping would have taken place. Additionally, any of the n operational batteries in the fleet can undergo substitution (Z), and be replaced by a new one.

$$M = \{\mu_1, \cdots, \mu_{\bar{M}}\} \tag{3.5}$$

$$\mu_k = \{Z_{1k}, \cdots, Z_{nk}, G'_k\} \tag{3.6}$$

$$Z_{ik} \in \{0, 1\} \text{ for all } i = 1, \cdots, n$$

$$G'_k \in \{1, \cdots, n_G\}$$

Therefore the number of possible actions ($\bar{M}$) will be $2^n \times n_G$.

3.2.3 Immediate Costs

Costs are attributed in this problem formulation to the actions mainly. However, for the swapping as we compare the new system placement with the older one to pinpoint the swapping costs, the cost dependence will include the current state as well. Swapping actions incur expenses associated with labor work and penalties for

potential of loss in fleet's output work. Similarly, substitution will incur costs associated with the new battery cost, labor work, and potential of loss. The costs incurred at each decision can be found by:

$$C(\varsigma_p, \mu_k) = \sum_{i=1}^{n} \{c_2 Z_i\} + c_{1,G_p,G'_p}\{G_p \neq G'_p\} \tag{3.7}$$

Where c_2 is the substitution cost coefficient, and c_{1,G_p,G'_p} is the cost coefficient associated with all the swapping actions necessary to fulfill the change of the system placement from G_p to G'_p. The coefficient is a function of G_p and G'_p, because of the fact that there could be one single swapping action or several ones required to change the system placement. For example, the penalty associated with the potential loss of output work from the fleet is dependent on how many vehicles are involved in the swapping. Therefore, the coefficient can be defined to have different values based on that. We note here that ideally there are no costs or rewards associated with the states themselves, as the definition of the states include only operational range of the batteries. However, there are hidden costs associated with some actions that may evolve the health states of the batteries beyond the operational range (over the threshold). The details of these are explained in Section 3.2.5.

3.2.4 Transition Probabilities

The transition probabilities are associated by definition with the uncertainty in the model, specifically the random error (ϵ) in the deterministic prediction of the health state. ϵ represents a random variable with assumed known distribution. Furthermore, if the health state prediction model is sufficiently accurate, ϵ as a random variable can be assumed to justifiably have the following properties: zero mean value, and symmetric probability distribution around the mean value. Additionally, with this assumption, ϵ theoretically will have a discrete uniform distribution, where all elements of the finite set of potential values are equally likely (something like tossing

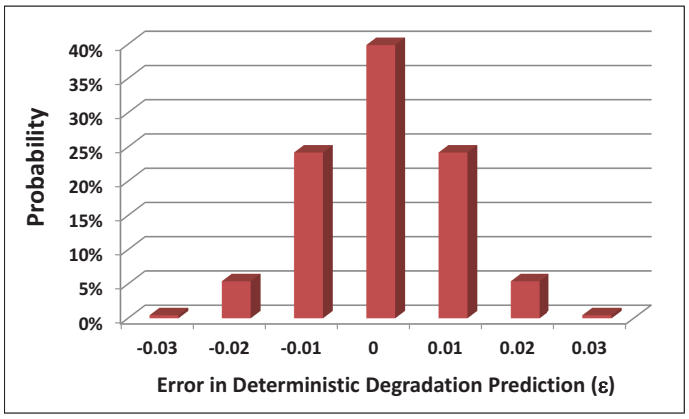


Figure 3.1: An Example of Probability Distribution of ϵ

a balanced coin or unbiased die, or the first card of a well-shuffled deck). However, in practice it is more likely to have higher probability attributed to the mean value and will have more of a bell shaped distribution. In fact, this would be the case if we use actual historic data to statistically derive the probability distribution. This hybrid theoretically-based practically-modified probability distribution will be used throughout the Chapter. Discretization of the probability to match the discrete SDP framework introduced above will result in probability distribution (probability mass function) similar to the one in Figure 3.1. In this figure, it is assumed that the number of possible values of ϵ is 7, symmetrically distributed around the mean, where higher probability is on the mean. We will refer to this number of possible values as the error resolution [see Section 3.3.4].

Practically, the exact values of the probabilities can vary depending on the application, the confidence in the deterministic health state predictor, and other miscellaneous factors. We show in Section 3.3.2 different probability distributions and its effect on the performance of the stochastic DBOS policy.

A final assumption which addresses the joint probability distribution is made. For simplicity, we assume that the random variables ϵ_i; for $i = \{1, \cdots, n\}$ are independent of each other. This is to say that the error in the deterministic prediction for any of the vehicles will be independent of the other errors in the predictions of the

health state evolution in the other batteries within the fleet. This assumption is not intuitive because of the nature of the fleet operation and Equation (2.3). However, we had assumed in our modeling that the health state predictor is accurate enough, and the error in the prediction is independent from the placement of the battery within the fleet, isolating the error as an independent random variable representing the uncertainty from other variables under study. Therefore, the assumption of the errors being independent of each other in this case becomes plausible. With this assumption, the joint probability distribution can be found by the multiplication rule for statistically independent random variables:

$$Prob(\bigcup_{i=1}^{n} \epsilon_i) = \prod_{i=1}^{n} Prob(\epsilon_i) \tag{3.8}$$

Now using the one step transition probability formulation from [81]:

$$\begin{aligned} Prob(\varsigma_i, \mu_k, \varsigma_j) &= Prob(\varsigma_j | \varsigma_i, \mu_k) \\ &= \sum_{\epsilon_i \in \bar{\epsilon}} Prob(\epsilon_i) 1_{\{\varsigma_j = S^M(\varsigma_i, \mu_k, \epsilon_i)\}} \end{aligned} \tag{3.9}$$

where $\bar{\epsilon}$ is the set representative of all the possible outcomes of the random variable ϵ, $1_{\{L\}}$ is the indicator function defined by:

$$1_{\{L\}} = \begin{cases} 1 & \text{if the statemet } L \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

and S^M is the states transition function given in DBOS case by Equation(3.1). Equation (3.9) can be written in a more simplified form by incorporating Equation(3.8)

:

$$
\begin{aligned}
Prob(\varsigma_i, \mu_k, \varsigma_j) &= Prob(\varsigma_j | \varsigma_i, \mu_k) \\
&= Prob(\epsilon_1 = \Lambda_1, \cdots, \epsilon_n = \Lambda_n) \\
&= Prob(\epsilon_1 = \Lambda_1) \cdots Prob(\epsilon_n = \Lambda_n) \qquad (3.11)
\end{aligned}
$$

where $Prob$ is the discrete probability found from the probability distribution supplied with the problem as a parameter (e.g., Figure 3.1), and $\Lambda_{i'}$ is the error value in the health state prediction for the battery (i') in the fleet that matches $Y_{i'i}$ (from state ς_i) and $Y_{i'j}$ (from state ς_j) given action μ_k. To clarify this more, we present the following example for a two-vehicle scenario. Assume the initial health states at the start of this interval were 0.06 and 0.11 for vehicles 1 and 2's batteries; respectively. Furthermore, assume the decision (action) taken at this interval is to opt out of any substitutions, and to place the batteries in loading profiles with deterministic degradation rates equal to 0.07 and 0.03, respectively. If we assume similar probability distribution to Figure 3.1, then there are 49 (from 7^2) possible outcomes for the batteries health states at the end of the current interval. The probability of the next state to have health states equivalent to 0.12 and 0.14 is equal to $Prob(\epsilon_1 = -0.01) \times Prob(\epsilon_2 = 0) = 0.242 \times 0.399 = 0.097$.

3.2.5 Hidden (Random) Costs

In many applications the one-period contribution function is a deterministic function of ς_i and μ_k , and hence written as the deterministic function $C(\varsigma_i, \mu_k)$. However, these costs do not fully represent the case here. There are random hidden costs in this problem formulation that will result in the case of some decision evolving the health states of the batteries (and in turn the SDP states) to a value beyond the operational range, i.e. undefined SDP states. The batteries in this sense are overused and hence

a suitable penalty should be incurred.

There are two main issues associated with this scenario. The first is the undefined states. The second is concerned with how to attribute this additional cost while formulating the policy as the decision maker at the start of the interval lacks information about the real outcome of the health states.

The problem with undefined states lies in the fact that consulting the policy after arriving at these states will no longer offer any decisions for the decision maker. Additionally, during the construction of the policy, we will be unable to progress to the next interval as these states are undefined and hence their evolution is undefined as well.

In some applications, extending the defined states range can solve the problem. However this will increase the state space size, which in some applications such as DBOS is highly undesirable, due to the originally large size. In other SDP frameworks, undefined states can be avoided by assigning zero probability to the action that may result in such state. However, in this problem it is not suitable to opt for this solution. The most significant reason behind this can be explained with the help of the example in Figure 3.2. In this example, the system has state equal to S_1.There are only two feasible actions: a_1 with the solid line arrow is "do not substitute" and a_2 with the double-lined arrow is "substitute". From the figure, it can be seen that there are 6 different possible outcomes (states). While a_2 evolves the state S_1 to any of the 3 defined possible states (S_4, S_5, S_6), a_1 has a 5% probability of evolving S_1 to the undefined state S_u. Clearly, a_1 will result in lower direct costs than a_2. If we are to assign zero probability for a_1 to avoid the undefined state, we will be missing on the chance of taking a more economic decision "fearing" a probability of 5%. Therefore this fix is not recommended.

The second issue with this scenario is the attributed costs because for the overusage of batteries. As the policy is formulating, the decision maker is not allowed (by

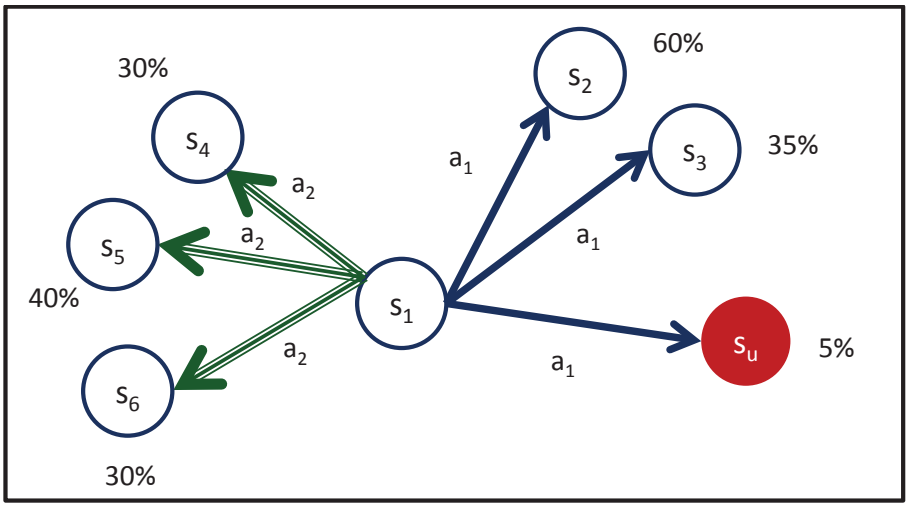


Figure 3.2: Undefined State Issue in DBOS

definition) to foresee the actual outcome of the new state. The health states can be within operational range or can represent an over-usage. In standard SDP framework, the recommended state dependent costs when the output state is random can be solved by the consideration of the expected value function[81]. The contribution of this hidden cost can be viewed as the expected contribution given that we are in state ς_t and take action μ_t. Using the discrete probability distribution and the compatible expected value correlation, the expected value of the state dependent cost in DBOS can be found by:

$$C_{\text{hidden}}(\varsigma_i, \mu_k) = \sum_{j=1}^{n_{\text{next}}} Prob_j(\varsigma_j |\, \varsigma_i, \mu_k) U(\varsigma_j) \tag{3.12}$$

where $U(\varsigma_j)$ is the added cost for being in state (ς_j). In DBOS case, this cost will be in the form of a penalty if the state is undefined. It means that $U(\varsigma_j)$ will be zero as long as we are in operational range of the health states, and will have a proportional penalty when batteries are overused. Equation (3.12) can be written as:

$$C_{\text{hidden}}(\varsigma_i, \mu_k) = c_{\text{hid}} \sum_{j=1}^{n_{\text{next}}} Prob_j(\varsigma_j |\, \varsigma_i, \mu_k) 1_{\{\varsigma_j \notin S\}} \tag{3.13}$$

where c_{hid} is the hidden cost (penalty) coefficient. While this approach solves the

second issue, the states (ς_j) are still undefined when we are outside the operational range.

We heuristically solve this by the following approach which specifically works for the DBOS SDP framework. It is expected that the decisions taken at the SDP states that include the upper limit of the operational range of the health states of the batteries to incorporate substitution due to the penalties incurred for over-usage. We will refer to these states as "border states". With respect to the policy itself, if some action will evolve the state to some undefined next state (in DBOS, it is an over-usage state), then we can use the closest defined state as a replacement to maintain the functionality of the policy for the next interval. This option maintains optimality if and only if the expected (intuitive) decisions for both the defined and undefined states are identical, and their expected next states are the same. In DBOS, the closest defined state to the undefined state is a border state. As these border states have optimal decision to substitute, and similarly the optimal decision (by intuition) at the undefined state is to substitute, then both have identical decisions. As the substitution in DBOS, by construction, will reset the battery health state, then the next state will not be a function of the previous health state for both defined and undefined states. Both decisions and next interval states are identical and hence optimality will not be jeopardized, and the policy can maintain functionality. To impose the penalty, the cost function will include the hidden cost that will be proportional to the probability of ending in an undefined state, similar to Equation (3.13). An example of this can be seen in Figure 3.3, where the border state here is S_3. With respect to the policy, the state evolution occurs as in the right hand part of the figure. The undefined state is not existent. Instead the 5% probability in action a_1 will be directed to the closest border state, S_3. Hidden cost will be incurred with this 5% probability.

One final remark about this heuristic approach is that the penalty will be imposed during the policy construction. However, during the policy validation, the penalties

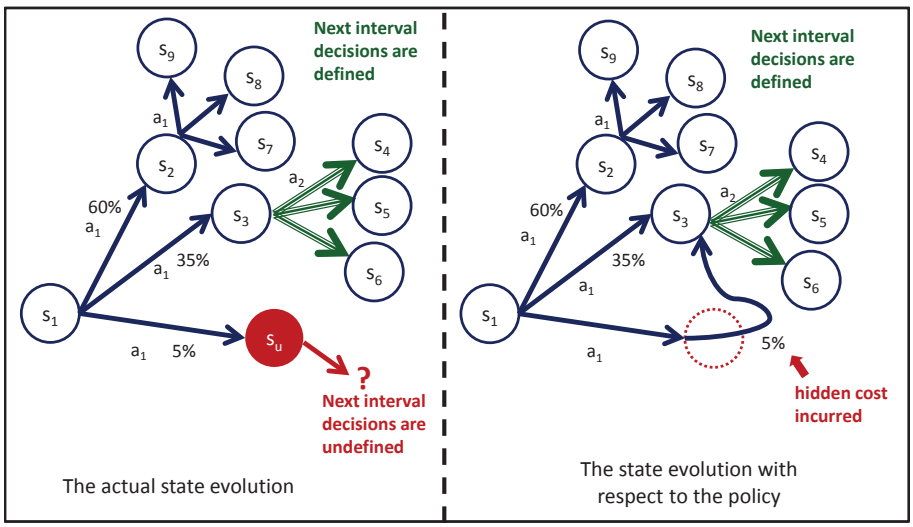


Figure 3.3: Heuristic Fix for Undefined States in DBOS

are only incurred after we identify the real next state and will be imposed if the state(s) have actually overstepped the operational range of the battery's health states. While this is not standard validation methodology, we explain the motivation behind this through the following analogy. A parent wants to motivate his middle school son to study more offering rewards if he gets high grads and threatening punishment (e.g., take his video games, grounding) if he does not achieve good grades (This is identical to the savings (rewards) for not conducting substitution vs. penalty costs (punishment) associated to overusing the batteries). Now the son has an exam next day. The exam requires 4 hours of study to fully cover the material (this is similar to doing all potentially necessary substitutions to avoid overusage). The exam is stochastic in nature. The teacher can bring all questions from specific part of the material or it can be distributed. The exam can be easy, or hard (This is similar to the degradation stochastic nature). The son saw the neighbors kids outside playing after a long winter and he is tempted to join them (this is similar to DBOS tempted to avoid substitutions). Now the son, based on the incentives (rewards and punishments) from his parent, formulates his own policy based on his intuition. Instead of studying the whole 4 hours, he decides to study for 2.5 hours as he feels he will be able to cover

with sufficient probability the material needed to get a good grade (this is similar to DBOS formulating the policy based on prediction and potential of overusage). Now, the parent has no idea and cannot actually punish the son until the exam comes, the stochastic part of it becomes known and the son either gets lucky (get good grade based on his limited covered material) or he gets unlucky (the exam turns out to be from material outside his coverage). After, the son gets his grade, only then the parent can actually reward or punish him. Similarly, during the policy validation, we have to wait until next state becomes known and the stochastic degradation is recognized. Only then we decide to punish the policy for overusing.

3.2.6 Solution Approach

Now that we have formulated the problem in the framework suitable for treatment using discrete stochastic dynamic programming, we solve this problem using backward induction dynamic programming. Backward induction minimizes a function of the following form [81]:

$$V^t(\varsigma^t) = \min_{\mu^t \in M^t} \left(C^t(\varsigma^t, \mu^t) + \sum_{\varsigma' \in S} P(\varsigma' | \, \varsigma^t, \mu^t) V^{t+1}(\varsigma') \right) \qquad \forall \varsigma^t \in S \tag{3.14}$$

where all superscripts indicate here the discrete time point in the horizon and ς' is a possible next state given ς^t. This function which represents the value of being at state ς^t is mostly known as the Bellman's equation. From its name, backward induction starts at the very end of the horizon and iterates towards the beginning, computing the value function $V^t(\varsigma^t)$ for all states $\varsigma^t \in S$ each time. The terminal value $V^T(\varsigma^T)$ is assumed to be given.

With respect to DBOS, we will start by a simplifying assumption: the terminal value at the end of the horizon is zero. It can be easily incorporated if we wish to sell

the batteries for example to a second market at the end of the horizon. Then, we will only need to know the expected revenue generated at that instant by selling these used batteries. With this, we will follow the Backward Induction procedure described below [81].

Step 0. Initialization

Initialize the terminal contribution $V_T(\varsigma_T)$

Set $t = T - 1$

Step 1. Calculate:

$$V^t(\varsigma^t) = \min_{\mu^t \in M^t} \left(C^t(\varsigma^t, \mu^t) + \sum_{\varsigma' \in S} P(\varsigma' |\, \varsigma^t, \mu^t) V^{t+1}(\varsigma') \right) \qquad \forall \varsigma^t \in S$$

Step 2. If $t > 0,$ decrement t and return to Step 1. Else, stop

3.3 Case Studies

In this section, we will demonstrate the strong performance of the stochastic DBOS policy versus the deterministic DBOS policy through a series of case studies. The solutions for the deterministic cases have been obtained through the application of the DBOS-specific Branch-and-Bound-based algorithm detailed in Section 2.3.2. The deterministic DBOS policy has already demonstrated the ability to outperform many common fleet joint maintenance-management policies as explained in details in the same reference. Therefore, we will only compare deterministic and stochastic DBOS policies. Although these case studies reflect small fleets, in principle the policy applies for larger fleets. With larger fleets, Approximate Dynamic Programming (ADP) techniques become essential due to the large state-space size (the curse of dimensionality) [81]. The details of how to augment these algorithms to DBOS scenario represent a

separate contribution and are beyond the scope of this thesis. Nonetheless, a short description of the opportunities to implement them will be highlighted in the Future Work Section in the final Chapter.

3.3.1 Case Study III

In this section, we report numerical results of a 3-vehicle fleet case study. The problem parameters are available in Table 3.1. The cost coefficients are inspired by real applications. The degradation coefficients have been modified to reflect shorter chosen plan horizon for the numerical case study as a sample problem. The modification in the coefficients is intended to simulate the real scenario where longer horizons are chosen, and thus substitutions are inevitable. The probability mass function of each of the errors in the deterministic health state prediction is similar to the one shown in Figure 3.1. Both deterministic DBOS and stochastic DBOS policies have been used to manage the fleet.

Table 3.1: Case Study III Parameters

Parameter	Symbol	Value
Number of vehicles	n	3
Number of loading profiles	m	2
Plan horizon (years)	K	5
Vehicles allocated per loading profile	d_j	[2,1]
Degradation rates (per month)	r_j	[0.05,0.08]
Swapping costs ($)	$c_1(k)$	800
substitution costs ($)	$c_2(k)$	11600
Threshold	β	0.2
Discretization interval (year)	Δ	1
Hidden cost (penalty) coefficient used in policy build-up*	C_{hid}	58000
Proportional penalty of over-usage of batteries used in policy validation ($/0.01 of over-usage in health state)		3000

* This number will be multiplied by a small probability as seen from Equation (3.13)

To mimic real life scenario, a random number between 0 and 1 is generated at

each discrete interval. Pinpointing the generated random number on the cumulative distribution, the next SDP state (and in turn the health states of the batteries) out of the 343 possible next states is selected. Due to the stochastic nature of the problem, the number of runs conducted per policy is 5000. All runs started with states with initial condition of brand new batteries and different initial system placements. When comparing the mean value of the estimated total maintenance costs for all the runs, it was found to be 37515 and 35001 for deterministic and stochastic DBOS, respectively. The improvement is only 6.7%, but this is not "the full picture" of the performance. This is mainly due to the fact that the mean value is zero, bringing the two policies mean performances close. A good representation of the performance would be in dividing the costs into several ranges and finding the frequency of occurrences as in Figure 3.4. It should be noted that the ranges on the figures are not equally divided, where we decrease the ranges around high frequency occurrences to provide deeper insights.

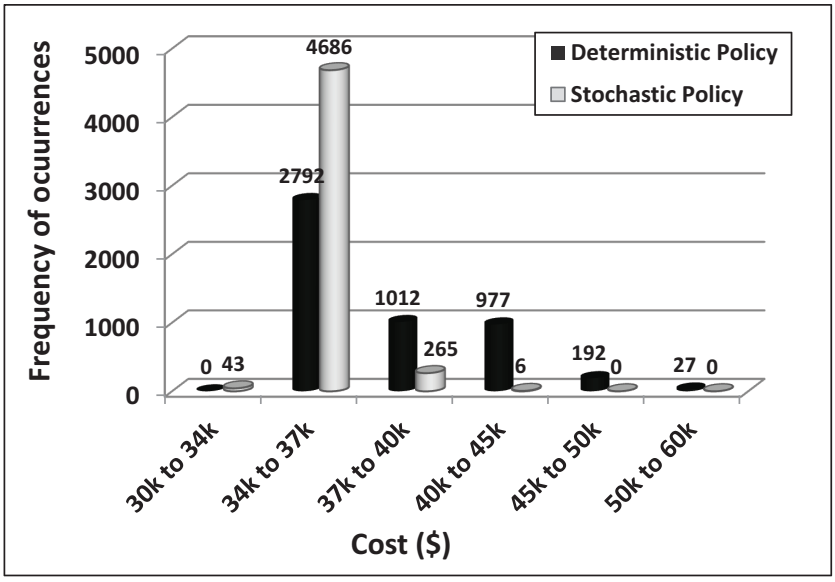


Figure 3.4: Case Study III Results

From Figure 3.4, it can be clearly seen how stochastic DBOS outperforms deterministic DBOS. Around the mean value, we find 93% of the stochastic policy runs

have achieved this range, in comparison to less than 55% of the deterministic runs. Additionally, in the higher ranges (above $40k), we find significant number of occurrences for the deterministic policy in comparison to extremely small number of occurrences for the stochastic policy. This reveals the inability of the deterministic policy to cope with "bad" scenarios where higher than predicted degradations in the batteries health states are taking place. Both (around the mean and higher ranges results) provide the conclusion that the stochastic DBOS is more robust in terms of handling uncertainties in the degradations and maintaining the maintenance plans costs at low levels.

3.3.2 Case Study IV

The second case study aims to examine the performance of the stochastic DBOS policy when applied to various error distributions. We maintain the assumption of the general shape of the error distribution in terms of symmetry around the mean, and a mean of zero error. However, the distributions standard deviations (and hence the variations) are varied from the original values showed in Figure 3.1. Table 3.2 shows the different distributions examined, where Dist C represents the distribution from Figure 3.1 and Dist F represents the theoretical uniform (equal probability) distribution.

Table 3.2: Different Error Probability Distributions Tested in Case Study IV

Distribution	Variance	Standard Deviation
Dist A	0.545	0.738
Dist B	0.734	0.857
Dist C (normal)	0.995	0.998
Dist D	1.260	0.122
Dist E	3.000	1.732
Dist F (uniform)	4.000	2.000

An increase in the variation reflects an increase in the uncertainty of the predicted error. More importantly, an increase in the variation reflects in the DBOS policy an

increase of the probability of larger errors (e.g., ϵ =0.02 and 0.03), and hence it would be expected that the policy will suffer a decline in the performance and an increase in the cost values, which will be observable in both stochastic and deterministic DBOS policies. Figure 3.5 reveals this fact. In the case of Distributions A through D, there is a gradual decline in the performance of the stochastic policy noted through the decrease of occurrences in the \$34k-\$37k cost category. The decline increases significantly through Distributions E and F. This is mainly attributed to the variance value jump increasing from Dist. D to Dist. E and finally Dist F. A similar behavior is noticed in the deterministic DBOS performance, though we note that stochastic DBOS policy is more adversely affected by the distribution variance. In fact, just comparing occurrences in the \$34k-\$37k cost category, the deterministic DBOS becomes very comparable to the stochastic DBOS policy in large variances relatively. However, upon examining the higher ranges (above \$40k), stochastic DBOS attributes can be easily recognized in minimizing the occurrences in these ranges. With high variance, deterministic DBOS shows increased occurrences in the higher ranges due to its inability to cope with the errors in the deterministic predictions.

The main conclusion of Case Study IV is that while stochastic DBOS performance intuitively deteriorates with increased variance, yet it maintains the ability to handle "bad luck" runs much better than deterministic DBOS, avoiding excessive maintenance costs.

3.3.3 Case Study V

The main objective of the third case study in this chapter is to examine the performance of stochastic DBOS policy formulated based on a specific distribution, when the exogenous information (error distribution) turns out to be different. In many cases, the true distribution of the prediction error cannot be known prior to actual implementation, or might be estimated mistakenly. Therefore, in this case

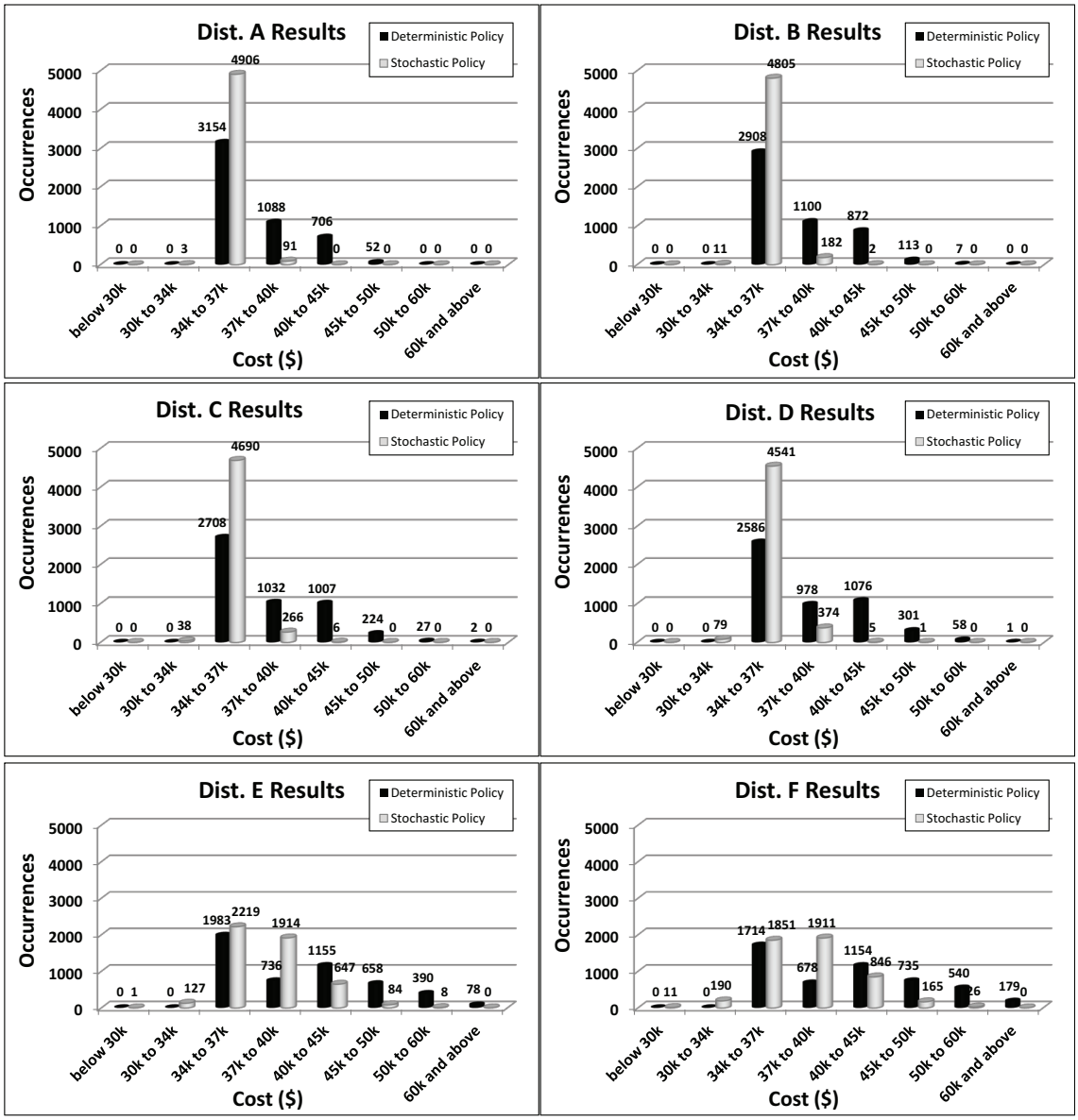


Figure 3.5: Case Study IV Results

study, we aim to investigate the loss in performance if the policy was build upon wrong distributions. The policies formulated in case study VI will be tested against distributions different from which they have been formed with. In specific, policies formulated by using Dist A and Dist C will be tested against Dist F, and vice versa.

3.3.3.1 True Exogenous Information = Dist C & Dist A

We first test stochastic DBOS policy formulated with Dist F against exogenous information with Dist C and Dist A. Figure 3.6 shows the results compared with the stochastic DBOS policy formulated with the correct (matching) error distributions and deterministic DBOS.

The results show a quite intuitive behavior. Using the correct distribution to build up the policy generates better results (the occurrences in the \$34k-\$37k category). Nonetheless, using stochastic DBOS policy even if it is build on wrong distribution outperforms deterministic DBOS. While deterministic DBOS becomes comparable in the mid ranges, the stochastic DBOS policy build on Dist. F still manages to minimize the occurrences in the high cost ranges (above \$40k).

3.3.3.2 True Exogenous Information = Dist F

Now, we present the second part of case study V, where the exogenous information has Dist F. We test stochastic policies formulated based on Dist A, Dist C, and the matching distribution Dist F. Results are shown in Figure 3.7. This case represents a counter-intuitive behavior as from the occurrences (especially in mid ranges), it can be noticed that stochastic DBOS policies formulated with Dist A and C outperform stochastic DBOS policy formulated with matching distribution (Dist F). The following behavior can be explained through the examination of the batteries health states evolution. The stochastic DBOS policies formulated with Dist A and Dist C tend to overuse the batteries (health states beyond threshold value) more frequently than

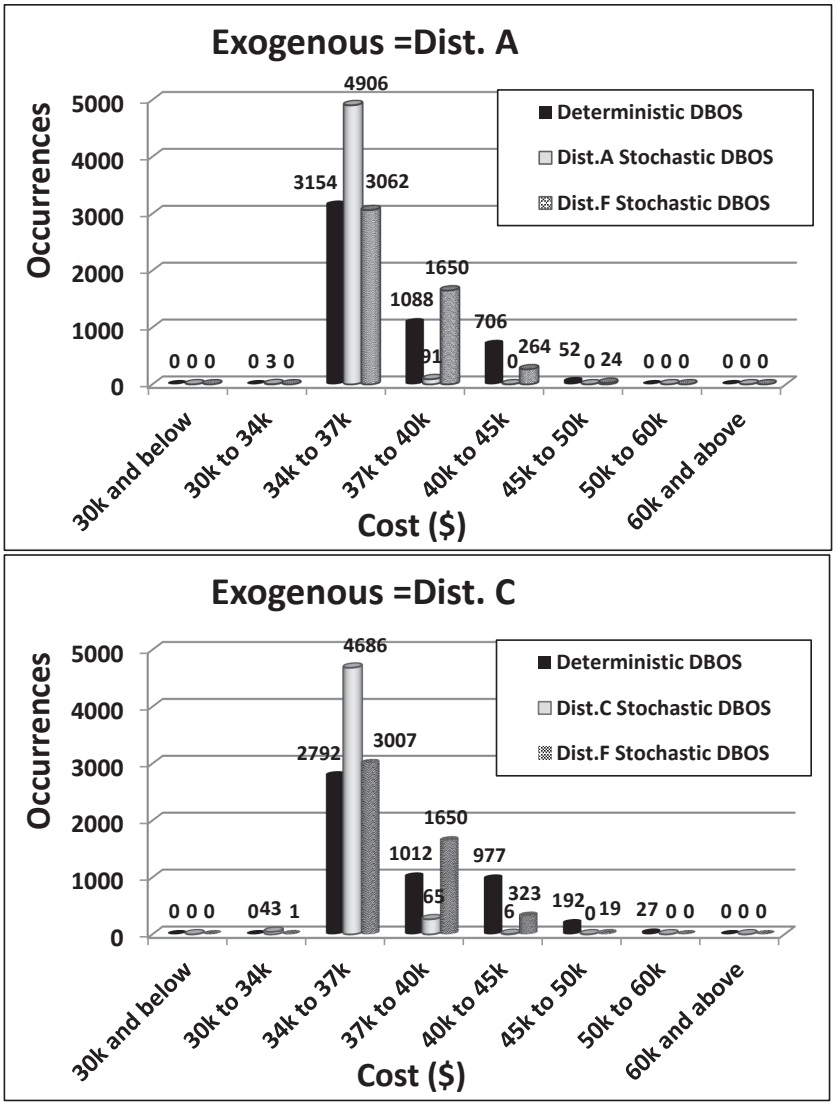


Figure 3.6: Case Study V- Part One Results

when the matching distribution has been used for policy buildup. Table 3.3 shows the occurrences of over-usage when different policies are used. It should be noted that this behavior for stochastic DBOS policies formulated with Dist A and C only occurs when tested on non-matching and harsh exogenous information environment. These policies themselves maintain safe factors similar to (or even better than) what Dist F based stochastic DBOS policy showed in terms of minimal over-usage. Table 3.4 shows that.

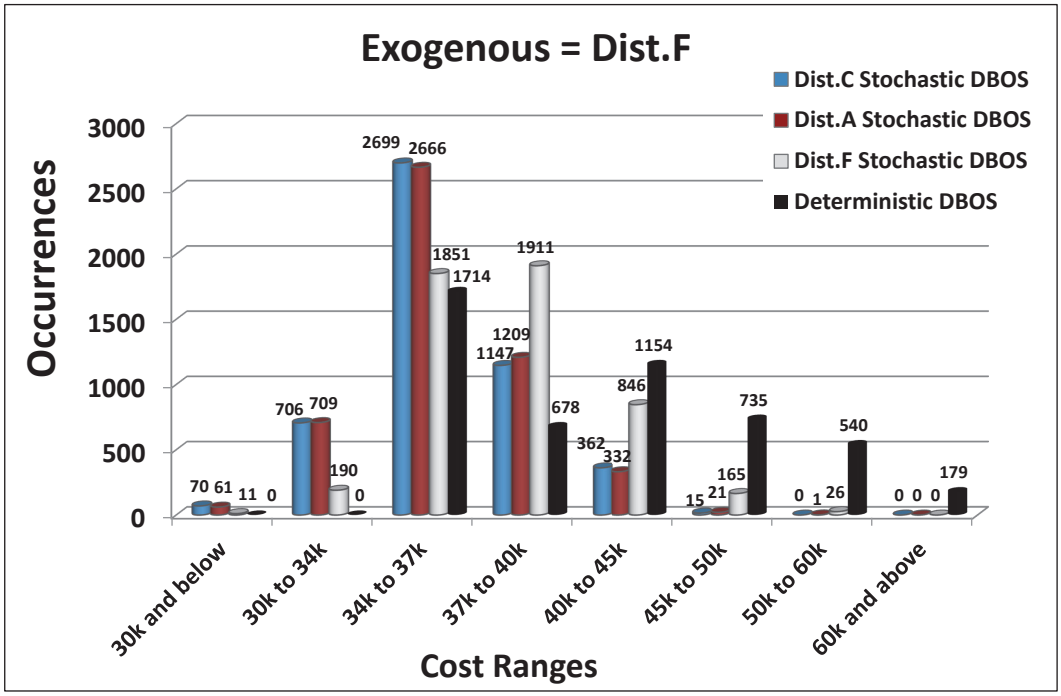


Figure 3.7: Case Study V-Part Two Results

Table 3.3: Over-usage of Batteries in Case Study V-b Results

Policy Formulated Based on:	Battery Health State = 0.21 Occurrences	Battery Health State = 0.22 occurrences
Dist F	266	0
Dist C	1686	291
Dist A	1750	316

In conclusion, for high variance exogenous information, other policies show lower costs but higher over-usages. In all scenarios, using a stochastic policy is better than

Table 3.4: Original Over-usage of Batteries in Dist A and Dist C based Stochastic DBOS Policies when Tested on Matching Distributions

Policy Formulated Based on:	Battery Health State = 0.21 Occurrences	Battery Health State = 0.22 occurrences
Dist C	66	5
Dist A	13	1

deterministic policy, even if it is built upon wrong distribution. Recommendations from this case study can be summarized with the following: If the error distribution is known to be of high variance and over-usage is an issue (sensitive), using similar distribution for policy build up will be better. If over-usage is not a major concern, using normal Dist C for policy buildup will be better. If error distribution is unknown, using Dist C provides the best tradeoff between low cost and reasonable number of over-usages.

3.3.4 Case Study VI

The objective of this case study mainly aims to examine the loss in performance when low error resolution is used. In specific details, we reduce error resolution in Figure 3.1 from 7 pillars (number of possible values taken by the random parameter) to 5 pillars and finally to 3 pillars, maintaining the general structural properties of the distribution. We maintain the mean of the error at 0, and maintain the sum of probabilities to be equal 1 by equally redistributing the canceled pillars. We use these new (low resolution) error distributions to formulate stochastic DBOS policies corresponding to them. Then we test the generated policies on exogenous environment that is of high resolution (7 pillars). The motivation behind this setup lies in the fact that stochastic DBOS policy (like most Stochastic Dynamic Programming instances) suffers from the curse of dimensionality. The reduction in the resolution of (ϵ) can significantly impact the necessary memory allocation and processing times (see Table 3.5). Therefore, if the loss in the performance due to lower error resolution is accept-

able, lower resolution stochastic DBOS policies can represent a more efficient option, computational-wise. Moreover, this scenario can be also similar to Case Study V motivation, where exact distribution of prediction error is not accurately attainable for physical or instrumental (measuring) reasons, and we would like to investigate the DBOS performance under such circumstances.

Table 3.5: Memory Allocation and Time Required for Different Resolutions of Dist. C based Stochastic DBOS policies

Policy	Max Memory Used	Time
Dist C - 7 Pillars Policy	3.5GB	2.3 hrs
Dist C - 5 Pillars Policy	1.1GB	1.5 hrs
Dist C - 3 Pillars Policy	0.25GB	0.9 hrs

Figure 3.8 shows the results of deterministic DBOS and stochastic DBOS with high and low error resolutions. As it can be seen from the cost ranges, the performance of the policy has not been affected by lowering the error resolution. In fact, it looks as if it is getting better. However, upon investigation of the health states evolution of the batteries as in Case Study III, we find out more over-usages are occurring with lower error resolutions. Table 3.6 shows the statistics of the over-usages. It can be clearly seen that with lower error resolution, more occurrences of over-usages are taking place.

Table 3.6: Over-usages of Batteries in Stochastic DBOS policies with Lower Error Resolutions

Policy	Battery Health State = 0.21 Occurrences	Battery Health State = 0.22 occurrences
Dist C - 7 Pillars Policy	66	5
Dist C - 5 Pillars Policy	92	5
Dist C - 3 Pillars Policy	245	21

The main finding of Case Study IV is that using stochastic DBOS is better than deterministic DBOS, even if it is built upon lower error resolution. For Dist. C, 5 pillars and 3 pillars cases are very comparable to high error resolution, where the

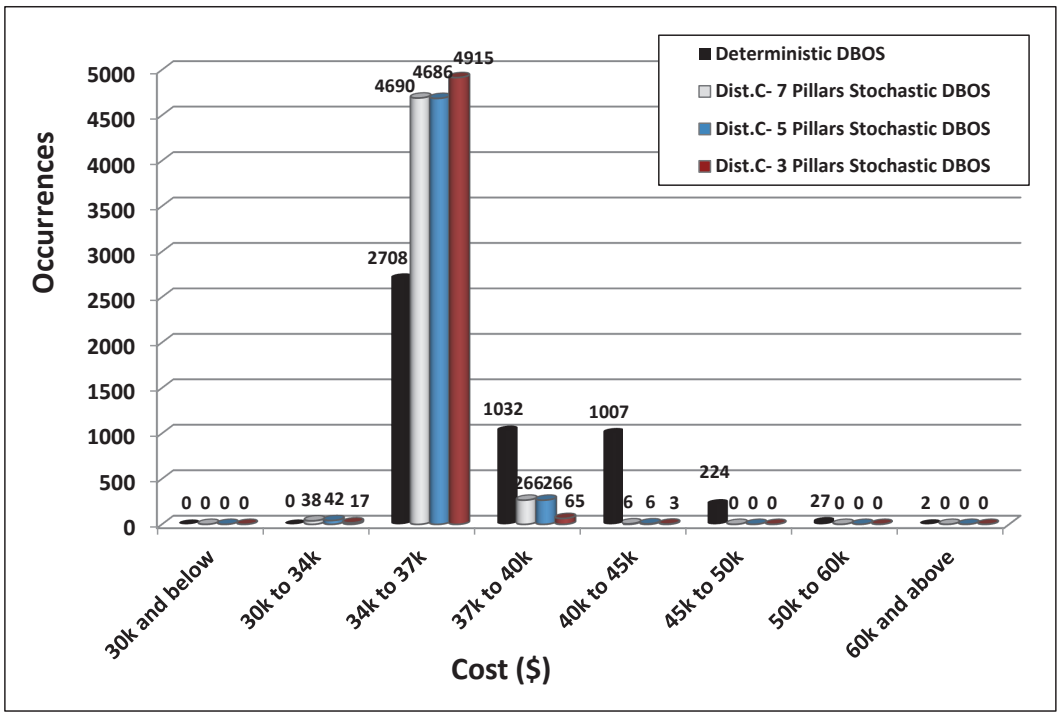


Figure 3.8: Case Study VI Results

lowest resolution (3 pillars case) showed more over-usages. Recommendations from Case Study IV can be summarized with the following: For Dist. C which represents practically the most likely error distribution, the use of 5 pillars low error resolution is justified.

3.4 Conclusions

We reformulated the degradation-informed resource allocation policy that we developed in the previous Chapters towards accounting for uncertainties in the batteries health states. Modifications to the deterministic model have generated several complications, which have been comprehensively solved for idealistic small fleets problems. Numerous case studies have shown stochastic DBOS to outperform deterministic DBOS. stochastic DBOS has been tested against variation in uncertainty, lack of knowledge of uncertainty distribution, and lower resolution of error. Results have shown stochastic DBOS's ability to avoid "bad luck" runs robustly, and hence avoid

excessive maintenance costs. stochastic DBOS has also outperformed deterministic DBOS in the cases where wrong distributions were used for the policy build-up. With stochastic DBOS policy (as most SDP instances) suffering from the curse of dimensionality and based upon results from Section 3.3, the decision support toolbox in Table 3.7 is generated. It represents the final recommendations with respect to best practice management for hybrid fleet management, in accordance to what has been developed.

Table 3.7: DBOS Decision Support ToolBox

Problem Size	Degradation Estimation	Recommended Policy
Small	High Confidence	Deterministic DBOS with modified SA algorithm
Large	High Confidence	Deterministic DBOS with DBOS-policy-specific B&B-based algorithm
Small	Uncertain, Error Dist. is Known, Sensitive to over-degradation	Stochastic DBOS (use matching dist. for policy buildup)
Small	Uncertain, Error Dist. is known, Not Sensitive to over-degradation	Stochastic DBOS (use Dist. C for policy buildup)
Small	Uncertain, Error Dist. is Unknown	Stochastic DBOS (use Dist. C for policy buildup)

CHAPTER IV

Degradation-based Optimal Swapping with Local Inventory

4.1 Introduction

Deterministic DBOS and stochastic DBOS have shown significant improvement in terms of maintenance management and fleet utilization, in comparison to other fleet management policies. However, both policies assume instantaneous new battery availability when substitution is to take place. In practice, batteries orders will be subject to several issues such as the lead time (which is the time between the placement of the purchase order and the delivery of the battery). For example, the requested items might not be in stock at the supplier's inventory. The loss of output work in fleets can be adversely effective in terms of loss of profits (as in delivery trucks), and might be in some applications unacceptable (as in public transportation fleets where there is no redundancy in the fleet). For this reason, it is customary for fleet owners (or whoever is in charge of fleet maintenance in case it is outsourced to another company) to acquire a local inventory for several spare parts, especially those known of frequent degradations and break downs. Not only this enables prompt maintenance actions without the need to wait for these spare parts to arrive in what is known as Mean Logistics Delay Time (MLDT), but also it can provide optimal purchasing policies

when the decision maker is well informed.

MLDT is defined as the average time that is required to obtain replacements from the manufacturer and transport them to the work site. In reactive maintenance, the MLDT can significantly affect the availability of the system by allowing the mean time between failures (MTBF) to acquire higher shares in the timeline. The availability of the system is a crucial indicator of the probability that the system maintains ability to operate as required during a target period. The operational availability can be found by calculating the percentage of the time when all the vehicles of the fleet are ready and operational, to the total time. If the fleet is not required to be operational at all times, the mentioned times will account only for periods at which fleet functionality is needed. It can be clearly understood that the total time includes when some vehicles are not operational (known as the downtime). Therefore, with no planning, downtime will consist of MLDT and the necessary time to restore the system to the target functionality after acquiring the necessary spare parts, known as Mean Time to Repair (MTTR).

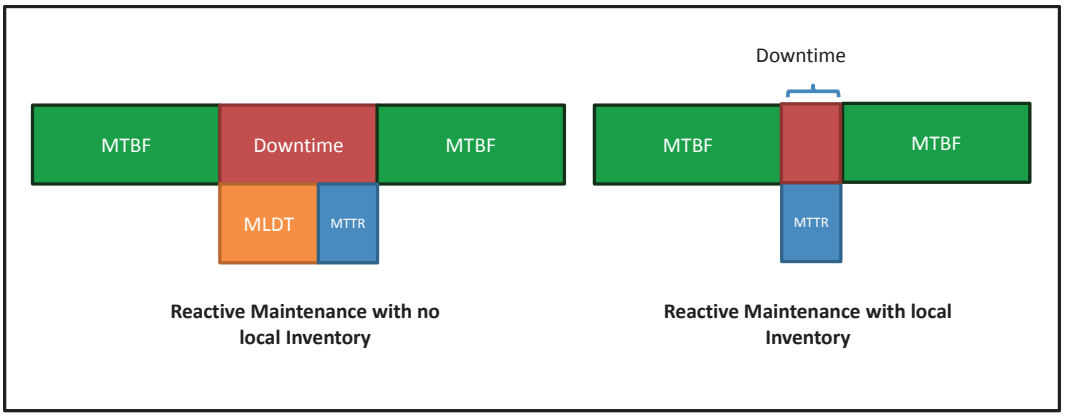


Figure 4.1: MLDT Effect on Downtime.

In fleet management, the MLDT can be avoided by the inclusion of the local inventory. The inclusion of local inventories requires special attention to the fleet maintenance policies. We established in previous Chapters the necessity of evolv-

ing the standard fleet joint maintenance-management policies, when hybridization (or electrification) is taking place due to the inclusion of the costly and degradable components such as Li-ion batteries. Similarly, there are several challenges associated with establishing and managing local inventories in fleet companies, especially those pertaining replacement batteries. This chapter will address these challenges through the development of a joint fleet-inventory management policy, denoted as the Integrated DBOS-Inventory policy.

The rest of this chapter is organized as follows. Section 4.2 will review relevant research work to the problem and specifically detail the associated challenges. Section 4.3 will focus on the augmentation of the DBOS model to accommodate the inventory management. Section 4.4 will discuss several case studies, where the performance of the integrated DBOS-Inventory model is examined.

4.2 Literature Review

Inventory management spells the difference between corporate success and failure [82]. It is concerned with specifying the sizes (and in some cases the places) of the stocked items. It dictates the amount to be ordered, and the time of the order. The main trade off in inventory management arises from the inclination for demand satisfaction, and the costs associated with maintaining the inventory. With consideration of additional factors such as the order costs and the lead times, inventory management is essential for functionality and profitability. For example, Economic Order Quantity (EOQ), which represents one of the oldest classical production scheduling models, can specify the optimal quantity of order that minimizes the inventory holding costs and ordering costs. This in turn projects significantly on profits (or losses).

The policies associated with the inventory replenishment systems are the "fruits of labor" of inventory models. They are concerned with the most important functions of inventory management in terms of when should orders be placed to restock inventory

and how much should be ordered. These policies are inherently affected by the demand type (as explained above), and the supply issues such as economies of scale (for production and delivery), capacity limits (for production and delivery) and delays in replenishment [82]. A concise representation of these policies (if it exists) depends mainly on the mentioned factors above, and the inventory related costs comprising of replenishment related costs, holding costs, and stockout costs.

An example of these policies is the famous (s, S) and (s, Q) policies. The former one specifies an optimal quantity of inventory level (s), known as reorder point, under which a restocking is to take place. That means if the inventory level is higher than the reorder point, restocking will not be optimal. If the level is lower, restocking will take place to bring the inventory level to an upper optimal value (S). The (s, Q) inventory policy (also known as the reorder point, order quantity system) focuses on separating the reorder point (s) and the ordering quantity to handle stochastic demands successfully. In terms of replenishment policies, we refer to the frequency of the inventory observation (review), which can be continuous, or conducted periodically at predefined intervals. A famous inventory policy compatible with the latter one is the (R, S) inventory policy, also known as periodic review policy. In this policy, the inventory level is only observed at intervals of R and if the inventory is at level (y), a quantity (S - y) is ordered to bring the inventory position to S [83].

In addition to the standard considerations in inventory management mentioned above, the battery replacements inventory will incorporate further challenges. The main challenge pertains to the fact that batteries are degradable even when they are stored at inventory. Spotnitz [84] reported that the loss of the Li-ion capacity in storage will have a typical shape like the one in Figure 4.2. This capacity loss exhibits reversible and irreversible components where only part of the capacity loss can be recovered by charging the cell. This small irreversible degradation in batteries can be significant when considering the fleet maintenance plan horizons. Hence, these

inventories are perishable or deteriorating inventories.

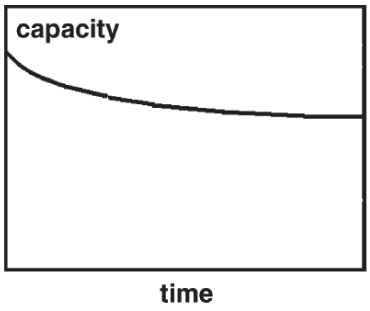


Figure 4.2: Typical Capacity vs. Time for Storage Test [84].

Deterioration is defined as damage, spoilage, decay, obsolescence, evaporation, pilferage, etc. that result in decrease of usefulness of the original one [85]. Most of the existing inventory models in the literature assume that items can be stored indefinitely to meet the future demands. However, certain types of commodities either deteriorate or become obsolete in the course of time and hence are unstable [86]. For example, the commonly used goods like fruits, vegetables, meat, foodstuffs, perfumes, alcohol, gasoline, radioactive substances, electronic components, etc., where deterioration is usually observed during their normal storage period. Therefore, if the rate of deterioration is not sufficiently low, or the duration of storage can be significantly large (like in DBOS case), its impact on modeling of such an inventory system cannot be ignored.

We refer to the distinction between deterioration and obsolescence in perishable inventories as clarified in [86]. Obsolescence refers to items that lose their value through time because of rapid changes of technology or the introduction of a new product by a competitor. Referred to as style goods, these items must be sharply reduced in price or otherwise disposed off after the season is over. An example of style goods are fashion goods [87] and spare parts for replaced models [88]. Electric (or hybrid) vehicles batteries are issued in generations, each of which partially eclipses the previous ones in terms of performance and compatibility. Nonetheless, it will hardly

render it obsolete. This is to say that older generations of batteries will remain useful as they are compatible to the already hybridized fleets. However they will not sustain the same value in the market as new generations arise (depreciation). While we investigate the influence of this behavior on the replenishment policies generated by the integrated DBOS-Local Inventory model through later case studies, the model itself is focused (perishable-wise) on the deterioration of batteries while in storages.

First efforts in perishable inventories included the fashion goods deteriorating at the end of prescribed storage period in [87] and the exponentially decaying inventory developed by Ghare and Schrader [89]. The exponentially decaying inventory refers to certain commodities shrinkage with time by a proportion which can be approximated by a negative exponential function of time. Since then, considerable work has been done in this field. Thorough reviews can be found in [90, 91, 86]. A less thorough review with more recent efforts is provided by [92].

Nahmias [90] categorized the deteriorating inventory models on the basis of shelf-life characteristics into fixed lifetime as in [93, 94], and random lifetime as in [95, 96, 97, 98, 99]. The former category included those cases where the lifetime is known a priori to be a specified number of periods or a length of time independent of all other parameters of the system. The latter category included the exponential decay as a special case and those cases where the product lifetime is a random variable with a specified probability distribution [90]. This categorization was an extension of Van Zyl's [100] one, where he used the classification "age dependent" perishability and "age independent" perishability to distinguish between fixed life and exponential decay. Goyal [86] extended Nahmias's [90] categorization to a third group of models in which the inventory decays corresponding to the proportional inventory decrease in terms of its utility or physical quantity.

The demand plays a key role in inventory modeling. It can be deterministic which includes steady, intermittent, and time variant (with a trend or seasonality), or it can

be stochastic (unpredictable, random). With this key role, it can be easily concluded that each of the research efforts in deteriorating inventory had adopted demand assumptions that governed the modeling effort. For example, deterministic uniform demand was assumed in [101, 102, 103], and deterministic time variant demand was assumed in [104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 85]. Examples of efforts that assumed stochastic demand with known probability distribution are [95, 99, 115]. Examples of efforts that assumed stochastic demand with arbitrary probability distribution are [116, 117, 97].

Most of the previous efforts investigate the inventory as an independent part (at least management wise), isolating it from the other operational practices within the functional productive entity (a production company, a service provider, etc.). The real life inter-relations which an inventory is expected to have with the rest of the operations are mimicked through the modeling and estimation of the demand and supply. This decentralization is very plausible when the inventory represents the dominant function (as in supermarkets and commercial stores, wholesalers, energy storing operations in energy companies, etc.). However, in many other applications, the inventory represents only one unavoidable part of the complicated and integrated productive scheme. In these cases, integrating the inventory management with the other operations can be of significant economical impact. For example, Exxon Chemicals estimated that the adoption of planning integration techniques (which inherently incorporates inventory management) has led to an annual reduction in operating costs by 2% and operating inventory by 20% [118]. Similarly, DuPont has also noted that the integration of planning and scheduling played a part in reducing the working capital tied up in inventory from 160 to 95 million dollars for a polymers facility [118].

Chemical production planning is not the only field where inventory-planning integration has received interest. In manufacturing, there have been a spate of programs

developed by industry, all aimed at reducing inventory levels and increasing efficiency on the shop floor. Examples include conwip, kanban, just-in-time manufacturing, lean manufacturing, and flexible manufacturing [83]. In transportation planning, inventory management has been integrated with the vehicle routing problem in what is known as the inventory routing problem. As mentioned in Section 1.2.1, the vehicle routing problem aims to design a set of m minimum cost vehicle routes through n customer locations, so that each route starts and ends at a common location and some side constraints are satisfied [12]. Integrating the inventory management to this problem reflects on the consideration of adding the inventory associated costs to the objective function. Research efforts in this field covered deterministic and stochastic demands, examples of which can be found in [119, 120, 121, 122, 123, 124].

From all of the previous, the importance of integrating the inventory with planning and scheduling is quite evident. Therefore, we aim in this Chapter to introduce some incorporation of inventory modeling with DBOS, which will strengthen its deliverables in terms of being a robust battery-level fleet management policy. We note that DBOS does not show clear demand structure for the new batteries prior to the substitution action. Therefore, using an independent inventory model is hindered by the lack of the demand information (even for inventory models with demand of arbitrary distribution, the demand is still assumed to be representative and have certain structure). It is for this reason that we will introduce a special formulation that includes both DBOS and the inventory functionality, to implicitly capture the demand. The augmentation of the DBOS model aims to account for the inclusion of a local inventory with deteriorating (in storage) batteries. In addition to the fleet management functionality, an optimal replenishment will be another deliverable.

4.3 Problem Formulation

To integrate inventory management with DBOS, several changes are to take place on the presented model in Section 3.2. These changes will include the establishment of the age-dependent substitution variables (decision variables), purchase or replenishment variables (decision variables), and inventory variables (dependent variables). We first provide the mathematical modeling, and then explain the necessary changes in the SDP framework that we have setup in the previous Chapter. The case studies in this Chapter will be solved based on the SDP framework. Nevertheless, the provision of a concise mathematical model in Section 4.3.1 has several benefits. The mathematical model presents the reference to which the SDP framework can be built upon, especially that we are using Markov Decision Processes, where the definitions of states and actions might be misleading. Additionally, the mathematical model can be used for optimization purposes especially for the high confidence (deterministic) degradation estimation as in Chapter II. Furthermore, the model can be used for simulation investigations to analyze different scenarios where the SDP is intractable. Finally, a representative model is always a good asset for further development down the line.

4.3.1 Mathematical Modeling of DBOS-Inventory Integrated Policy

On the one hand, the placement aspect of DBOS, modeling wise, is not changed. Therefore, we maintain the placement variables $X_{ij}(k)$ with no change, and maintain the relevant constraints that involve these variables solely as in Equations (2.2), (2.3), (2.8), and (2.9).

On the other hand, the substitution variables $Z_i(k)$ require augmentation. Previously, these were binary variables that reflected the occurrence of a substitution action, at which a specific old battery is replaced with a new one. With the inclusion of the local inventory, and the decaying of batteries within the inventory, this

substitution may not incorporate a new battery with zero accumulative degradation. The battery that is replacing the degraded one might have been in storage for a while, and hence partially degraded. The substitution variable thus is no longer a mere flag of replacement, it has to represent the age of the battery that is replacing the degraded one. To accommodate this, we make a simplifying assumption. The batteries degradation while at inventory is relatively small. Moreover, we expect the inventory environment (in terms of temperature, humidity, etc.) to be well controlled. Therefore, it becomes plausible to assume the degradation of the batteries within the inventory to be of deterministic nature. This means that if we know the duration of time for which the battery has been placed at the storage, we can find its inventory deterioration. We augment $Z_i(k)$ as in Figure 4.3 to become an age-dependent substitution variable $Z_{iq}(k) \in \{0, 1\}$, where the subscript $q = 1, \cdots, q_m$, represents the age of the replacing battery with upper bound q_m. The upper bound can be found by:

$$q_m = \min\left(K, \frac{\beta}{\rho}\right) \tag{4.1}$$

where β, as defined previously, is the threshold value at which the battery becomes unfit to be in the field and requires replacement (substitution), ρ is the deterioration rate of the batteries in inventory and K is number of intervals found from the horizon plan (T) and the discretization interval (Δ).

Figure 4.3 shows an example where substitution has been scheduled for the second battery in interval 2, and the third battery in interval 4. On the left hand side, the substitution is age-independent as in previous chapters. On the right hand side, the substitution is age-dependent, where the replacing in the first substitution is conducted by a "fresh" new battery, and the replacing in the second substitution is conducted by a three-intervals-old (in inventory) battery.

In the DBOS-inventory integrated model, the substitution is separated from the

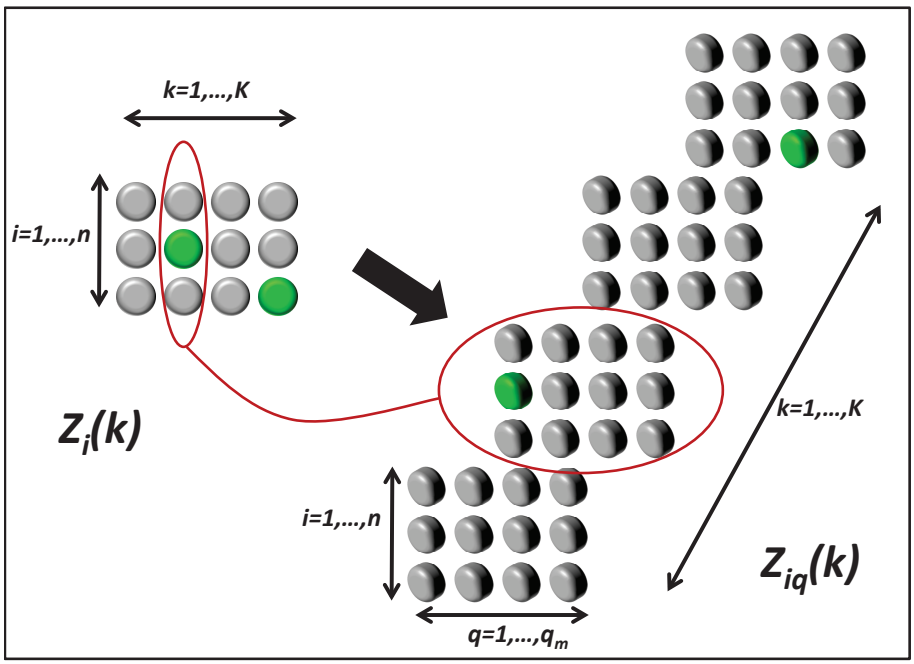


Figure 4.3: Augmenting the Substitution Variable

acquiring of new batteries (purchase). Therefore, we define a purchase (replenishment) variable $P(k) \in 0, \cdots, P^u$, which is a discrete decision variable that states the number of new batteries to be ordered at the beginning of each discrete interval. The upper limit P^u can be determined by a number of factors such as maximum purchase capability, maximum delivery size, supplier capacity, etc. With this construction, we are at most selecting one age-dependent substitution per battery. Therefore, we derive the following physically intuitive constraint:

$$\sum_{q=1}^{q_m} Z_{iq}(k) \leq 1 \qquad \forall i = 1, \cdots, n; \quad \forall k = 1, \cdots, K \tag{4.2}$$

Following next, we define the dependent variables $Q_q(k) \in 0, \cdots, Q^u$, which we denote as inventory variables. Inventory variables indicate the stock level of each possible age category, and are governed by the following equations:

$$Q_q(k) \geq 0 \qquad \forall q = 1, \cdots, q_m; \quad \forall k = 1, \cdots, K \quad (4.3)$$

$$Q_1(1) = Q_{initial} - \sum_{i=1}^{n} Z_{i1}(1) \quad (4.4)$$

$$Q_1(k) = Q_1(k-1) - \sum_{i=1}^{n} Z_{i1}(k) \qquad \forall k = 2, \cdots, \lambda \quad (4.5)$$

$$Q_1(k) = P(k-\lambda) - \sum_{i=1}^{n} Z_{i1}(k-\lambda) \qquad \forall k = \lambda, \cdots, K \quad (4.6)$$

$$Q_q(k) = Q_{q-1}(k-1) - \sum_{i=1}^{n} Z_{i1}(k) \qquad \forall q = 2, \cdots, q_m, \quad \forall k = 2, \cdots, K \quad (4.7)$$

$$Q_q(1) = 0 \qquad \forall q = 2, \cdots, q_m \quad (4.8)$$

where $Q_{initial}$ is the stock level of new batteries at the beginning of the plan horizon, and λ is the lead time between placing an order and its delivery. $Q_{initial}$ is assumed to be zero in most cases; nonetheless we maintain it as a parameter because in some cases the purchase contract some fleet companies strike incorporates spare parts with the main items in the contract (being in our case the hybrid or electric vehicles). Equation (4.3) is a physical constraint of the stock level being non-negative. Equations (4.4) through (4.7) govern the interrelation between the three groups of variables (age-dependent substitution, purchase, and inventory variables). Equation (4.8) states that there are only new batteries in stock in the first interval.

We finally reformulate the health state update in Equation (2.11) to incorporate the new age-dependent variables. The updated equation for the accumulative degradation variables $Y_i(k)$ is:

$$y_i(k) = \sum_{q=1}^{q_m} [(1 - Z_i(k))y_i(k-1) + f_q Z_{iq}(k)] + \sum_{j=1}^{m} r_j X_{ij}(k)$$
$$\forall i = 1, \cdots, n; \quad \forall k = 2, \cdots, K \tag{4.9}$$

where f_q is a vector that establishes the initial decay within the inventory (for each age category) for the replacing batteries. This vector can be either supplied as a parameter or found by:

$$f_q = (1 - q)\rho \qquad \forall q = 1, \cdots, q_m \tag{4.10}$$

where we assume linear decay within inventory. Bound constraints in Equation (2.12), and accumulative degradation initialization in Equation (2.11) require no changes, and can be used here:

$$0 \leq y_i(k) \leq \beta \qquad y_i(0) = 0 \qquad \forall i = 1, \cdots, n \qquad \forall k = 1, \cdots, K; \quad \forall i = 1, \cdots, n \tag{4.11}$$

To account for uncertainty, we use similar approach to the one explained in Section 3.2, where Equation (4.9) becomes:

$$y_i(k) = \sum_{q=1}^{q_m} [(1 - Z_i(k))y_i(k-1) + f_q Z_{iq}(k)] + \sum_{j=1}^{m} r_j X_{ij}(k) + \epsilon \tag{4.12}$$
$$\forall i = 1, \cdots, n; \qquad \forall k = 2, \cdots, K$$

where ϵ represents the error in the deterministic degradation prediction, identified after the discrete interval has passed.

The objective function has been selected previously to be in the form of minimizing total maintenance plan's projected costs. With inventory, this objective is comprised of Swapping cost, Purchase cost, Holding Cost, and Substitution cost. The swapping cost can be found similarly to Equation (2.13):

$$J_{swap} = \frac{1}{2}\sum_{k=2}^{K}\left\{c_{swap}(k)\sum_{i=1}^{n}\sum_{j=1}^{m}|X_{ij}(k) - X_{ij}(k-1)|\right\} \tag{4.13}$$

The purchase cost can be found by:

$$J_{swap} = \sum_{k=1}^{K}\{c_p(k)P(k) + c_{order}(k)(P(k) \neq 0)\} \tag{4.14}$$

where $c_p(k)$ is the time dependent cost of new battery, and $c_{order}(k)$ is time dependent order placement cost.

The holding cost for inventory can be found by:

$$J_{holding} = \sum_{k=1}^{K}\sum_{q=1}^{q_m}\{c_h(k)Q_q(k)\} \tag{4.15}$$

where $c_h(k)$ is the time dependent holding cost coefficient that can reflect a number of costs associated with maintaining the inventory environmentally (temperature, humidity), and operationally (inventory labor, renting space, electricity, etc.). The last contributing cost to the objective function is the substitution cost, which can be found by:

$$J_{holding} = \sum_{k=1}^{K}\sum_{q=1}^{q_m}\sum_{i=1}^{n}\{c_{sub}(k)Z_i(k)\} \tag{4.16}$$

where $c_{sub}(k)$ is the time dependent substitution cost coefficient reflecting the labor cost and potential of loss in work load. We note here that this coefficient is different from the one shown in Section 2.2.3 as it does not incorporate the price of new battery covered by the purchase cost.

The objective function then becomes now:

$$
\begin{aligned}
J =& \frac{1}{2}\sum_{k=2}^{K}\left\{c_{swap}(k)\sum_{i=1}^{n}\sum_{j=1}^{m}|X_{ij}(k)-X_{ij}(k-1)|\right\} \\
&+\sum_{k=1}^{K}\{c_p(k)P(k)+c_{order}(k)(P(k)\neq 0)\}+\sum_{k=1}^{K}\sum_{q=1}^{q_m}\{c_h(k)Q_q(k)\} \\
&+\sum_{k=1}^{K}\sum_{q=1}^{q_m}\sum_{i=1}^{n}\{c_{sub}(k)Z_i(k)\} \qquad (4.17)
\end{aligned}
$$

4.3.2 Integrated DBOS-Inventory SDP Formulation

We augment the $SDP = \{S, M, R, C, P\}$ formulation provided in Section 3.2, to account for the inclusion of inventory management. As the states in Markov Decision Processes (MDP) are assumed to store all relevant information necessary for the decision maker to take action, their definition should be extended with inventory. The extension has two case scenarios depending on whether a lead time exists or not.

4.3.2.1 States with No Lead Time

With no lead time to include, only the information regarding the stock levels of each age category requires incorporation in the state definition. Thus, we have:

$$S = \{\varsigma_1, \cdots, \varsigma_N\} \qquad (4.18)$$

$$\varsigma_p = (Y_{1p}, \cdots, Y_{np}, G_p, Q_{1p}, \cdots, Q_{q_m p}) \qquad p = 1, \cdots, n \qquad (4.19)$$

$$Y_{ip} \in \{0, \phi, 2\phi, \cdots, \beta\} \qquad \forall i = 1, \cdots, n$$

$$G_p \in 1, \cdots, n_G$$

$$Q_{qp} \in \{0, 1, \cdots, Q^u\} \qquad \forall q = 1, \cdots, q_m$$

where, as in Section 3.2.1, Y_{ip} represents the accumulative degradation of the

ith battery in state ς_p measure with discretization resolution ϕ, and G_p represents the system placement with n_G possible placements. The new variable in the state definition, Q_{qp}, represents the stock level of batteries of age q in the inventory. The total number of states (N) in the state space can be found now by $(\left(\frac{\beta}{\phi}\right)+1)^n \times N_G \times (Q^u)^{q_m}$.

4.3.2.2 States with Lead Time

Previously, with no lead time, our unconstrained access to batteries at anytime allowed room for freedom in terms when orders arrive and when they are used in substitution actions and when they are stored. Specifically, while all the previous occur during an interval, we had not pinpointed exactly when these happen within the interval. When there is lead time, we have to "tread carefully" in our definitions with respect to the exact moment purchase, substitution, and storage takes place. This is primarily due to the fact that actions have no access to the ordered batteries until the orders lead time has passed. We are then faced by two scenarios when there is lead time. The first occurs when the lead time is equal to one interval. The second occurs when the lead time is larger than that. These two scenarios arise from a chosen definition in terms of the states and actions, and their reflection on the real time horizon (pinpointing them in an interval). We set up the MDP such that the action is conducted exactly right after we become knowledgable of the states (battery health states and inventory levels). That includes the swapping, the substitution and most importantly the purchase. Therefore, the decision to purchase with lead time that equals one interval means that the next time we check the inventory levels, the item(s) we ordered are already delivered. In that case, the state definition requires no change. The difference from the zero lead time case will take place in the state transition as will be shown in Section 4.3.2.4.

When there is lead time that is larger than one interval, we will not find the orders

delivered the next time we check the inventory levels. In that case the orders will be "pending". The information about the purchase orders that have been placed but not yet delivered are essential for the decision maker. It becomes clear that the state definition in this second scenario requires their inclusion. The inclusion should be an efficient summary of the orders that have been placed, and their times. For this reason, we define a pending inventory variable ($Q_{pending\Lambda p}$) to store this information in. $Q_{pending\Lambda p}$ will state how many batteries have been ordered that require Λ intervals to be delivered. Intuitively, $Q_{pending1p}$ will state the order that will be delivered in the next interval. The state definition becomes:

$$S = \{\varsigma_1, \cdots, \varsigma_N\} \tag{4.20}$$

$$\varsigma_p = (Y_{1p}, \cdots, Y_{np}, G_p, Q_{pending1p}, \cdots, Q_{pending\lambda p}, Q_{1p}, \cdots, Q_{q_m p}) \quad p = 1, \cdots, n \tag{4.21}$$

$$Y_{ip} \in \{0, \phi, 2\phi, \cdots, \beta\} \qquad \forall i = 1, \cdots, n$$

$$G_p \in \{1, \cdots, n_G\}$$

$$Q_{pending\Lambda p} \in \{0, 1, \cdots, P^u\} \qquad \forall \Lambda = 1, \cdots, \lambda - 1$$

$$Q_{qp} \in \{0, 1, \cdots, Q^u\} \qquad \forall q = 1, \cdots, q_m$$

with $q_m = \min\left(T - (\lambda - 1), \frac{\beta}{\rho}\right)$ due to the arise of the $Q_{pending\Lambda p}$ variables that are now taking the place of the Q_{qp} variables. The total number of states (N), can be found now by $(\left(\frac{\beta}{\phi}\right) + 1)^n \times n_G \times (Q^u)^{q_m + \lambda}$.

4.3.2.3 Actions

The actions (decisions) definition has to be extended as well. It will comprise of all the possible combinations of swapping, age-dependent substitutions, and purchase (replenishment). Thus, we have:

$$M = \{\mu_1, \cdots, \mu_{\bar{M}}\} \tag{4.22}$$

$$\mu_k = \{z_{1k}, \cdots, z_{nk}, G'_k, P_k\} \tag{4.23}$$

$$z_{ik} \in \{0, 1, \cdots, q_m\} \qquad with\ no\ lead\ time\ and\ \forall i = 1, \cdots, n$$

$$G'_k \in \{1, \cdots, n_G\}$$

$$P_k \in \{1, \cdots, P^u\}$$

where z_i is a variable indicating whether a substitution has taken place or not for battery i and what was the age category of the replacing battery. For example, $z_i = 0$ represents no substitution for battery i, and $z_i = 6$ means a substitution has taken place with respect to battery i, where the replacing battery has been in inventory for 6 intervals. It is clear that z is a summarized representation of the age-dependent Substitution variable $Z_{iq}(k)$ described in Section 4.3.1. Therefore the number of possible actions ($\bar{M}$) will be $(q_m + 1)^n \times n_G \times P^u$.

With lead time, the same definitions apply except when q_m is selected from K (or $K - (\lambda - 1)$ when the lead time is larger than one). The difference is that z_i will run only for $i = 1, \cdots, q_m - 1$ as substitutions with batteries older than this cannot take place (time line wise) within the plan horizon (i.e. these specific sort of substitutions are offered after the plan horizon would have ended and by then no actions are needed).

4.3.2.4 Immediate Costs, Hidden Costs, Transition Probabilities, and Solution Approach

As in Chapter III, we will have two categories of costs associated with the actions taken and states arrived at. The immediate cost extension will include the inventory relevant costs detailed in Section 4.3.1. The new immediate cost equation becomes:

$$C(\varsigma_p, \mu_k) = c_{sub} \sum_{i=1}^{n} \{z_{ik} \neq 0\} + c_{1,G_p,G'_k} \{G_p \neq G'_{pk}\} + c_p\{P_k\} + c_{order}\{P_k \neq 0\} + c_h \left\{ \sum_{q=1}^{q_m} Q_{pq} \right\} \quad (4.24)$$

where c_{sub}, c_p, c_h, and c_{order} are defined in Section 4.3.1, and $c_{1,G_p,G'_{pk}}$ is defined in Section 3.2.3 in the previous Chapter.

Before talking about transition probabilities, we note that the transition of states is more complex in the DBOS-Inventory Integrated model. The transition of the first components in the state definition (battery health states and system placement) is similar to what has been described in Chapter III, with the health state evolution governed now by Equation (4.9) rather than Equation (3.1). However the evolution of the inventory related variables depends on whether there is lead time or not, since the state definition changes with that. Whether there is lead time or not, the transition should satisfy the rational of Equations (4.4) through (4.8). However, the difference in the state definitions in the two cases motivates the provision of detailed state transition equations. With no lead time, the state transition from ς_p to $\varsigma_{p'}$ with action μ_k is found by:

$$Q_{1p'} = P_k - \sum_{i=1}^{n} \{z_{ik} = 1\} \quad (4.25)$$

$$Q_{2p'} = Q_{1p} - \sum_{i=1}^{n} \{z_{ik} = 2\} \quad (4.26)$$

$$\vdots$$

$$Q_{q_m p'} = Q_{(q_m-1)p} - \sum_{i=1}^{n} \{z_{ik} = q_m\} \quad (4.27)$$

In the lead time case, once again we identify the two scenarios explained in Section

4.3.2.2. When the lead time equals one interval, the next state inventory level will be a reflection of the purchase action. However, the substitution actions accompanying the purchase action can no longer benefit from the purchase:

$$Q_{1p'} = P_k \tag{4.28}$$

$$Q_{2p'} = Q_{1p} - \sum_{i=1}^{n} \{z_{ik} = 1\} \tag{4.29}$$

$$Q_{3p'} = Q_{2p} - \sum_{i=1}^{n} \{z_{ik} = 2\} \tag{4.30}$$

$$\vdots$$

$$Q_{q_m p'} = Q_{(q_m - 1)p} - \sum_{i=1}^{n} \{z_{ik} = q_m - 1\} \tag{4.31}$$

When the lead time is larger than one interval, the $Q_{pending\Lambda p}$ variables place a buffer between the purchase action and the real inventory variables. Specifically, the purchase action will reflect only on $Q_{pending(\lambda-1)p'}$, and the real inventory variable $Q_{1p'}$ will involve only $Q_{pending1p}$:

$$Q_{pending\lambda-1p'} = P_k \tag{4.32}$$

$$Q_{pending(\lambda-2)p'} = Q_{pending(\lambda-1)p} \qquad applies\ when\ \lambda \geq 3 \tag{4.33}$$

$$Q_{pending(\lambda-3)p'} = Q_{pending(\lambda-2)p} \qquad applies\ when\ \lambda \geq 4 \tag{4.34}$$

$$\vdots$$

$$Q_{pending1p'} = Q_{pending2p} \qquad applies\ when\ \lambda \geq 3 \tag{4.35}$$

$$Q_{1p'} = Q_{pending1p} \tag{4.36}$$

$$Q_{2p'} = Q_{1p} - \sum_{i=1}^{n} \{z_{ik} = 1\} \tag{4.37}$$

$$Q_{3p'} = Q_{2p} - \sum_{i=1}^{n} \{z_{ik} = 2\} \tag{4.38}$$

$$\vdots$$

$$Q_{q_m p'} = Q_{(q_m-1)p} - \sum_{i=1}^{n} \{z_{ik} = q_m - 1\} \tag{4.39}$$

With respect to transition probabilities, there are no additional sources of uncertainty. The developed framework in Chapter III can be used with one small change, upper and lower bounds for Q_{qp}. In this case two options emerge based on the application and the fleet company preference. A strict lower and upper bounds can be imposed by assigning zero probability to any action that may overstep the operating range of the inventory levels. Unlike the case in Chapter III where we desired to maintain all possible actions due to the stochastic outcome of the battery health state evolution, the inventory evolution is 100% deterministic. Therefore, the next stocking levels will be known as soon as we make our decision, and hence the zero probability solution is effective.

The second option will emerge when there are more leniencies in the company's operation. For example with respect to the upper bound, the company can have

the ability of storing excess inventory at other places which will be more expensive intuitively. Another example is relevant to the lower bound. For example, we can define the case when Q_{ip} is dropping below 0, to be a representation of rush orders. In this case, the company can for example reach for an expensive local supplier (or local competitor) with zero lead time. It is undoubted that in this case, we will have to use the border state replacement procedure similar to the one detailed in Section 3.2.5 to maintain the functionality of the policy. The borders here are drawn with respect to the inventory level and not the health state. In both examples, penalties formulated by next-state dependent costs are added, and these costs can be added to the SDP. We finally reemphasize the note that while these are next-state-dependent costs, they are not hidden costs as the inventory evolution within the state evolution in this framework is deterministic. The framework is now suitable for treatment using discrete stochastic dynamic programming. We solve this problem using backward induction dynamic programming as described in Section 3.2.6.

4.4 Case Studies

In this Section, we will demonstrate the important role of the integrated DBOS-Inventory policy in terms of providing savings when stocking batteries for the fleet company is of importance. Although these case studies reflect small cases (two vehicle swapping), in principle the policy applies for fleets. The DBOS-Inventory framework is an extension of the DBOS one, which suffers as most SDP instances from the curse of dimensionality. This extension increases the state-space size and the severity of the curse. Approximate Dynamic Programming (ADP) techniques can be helpful in this manner. We reiterate here that the scope of the research work in this thesis is the establishment of the new and unprecedented DBOS and Integrated DBOS-Inventory policies, on the theoretical level. Therefore, details of how to augment the ADP algorithms represent a separate contribution and are beyond the scope of this research

work. Nonetheless, a short description of the opportunities to implement them will be highlighted in Section 5.3. Per the motivation of inventory explained in Section 4.1, stocking batteries is beneficial in two cases: non-zero lead time, and special pricing circumstances. Examples of the latter one is a special introductory price when spare batteries are part of the fleet hybridization (or electrification) contract, and ramp up prices (increase in the price of batteries with time). Taking this into consideration, the case studies aim to investigate the behavior of the Integrated DBOS-Inventory policy, to verify its compliance with the inventory motivation rational.

4.4.1 Case Study VII

The first case study in this Chapter aims to examine the Integrated DBOS-Inventory policy with a fixed purchase price (throughout the horizon) scenario and with zero lead time. The problem parameters are available in Table 4.4.1.

Due to the stochastic nature of the problem, the number of runs conducted per policy is 5000. All runs started with states with initial condition of brand new batteries and different initial placements. Figure 4.4 shows the results of stochastic DBOS (developed in Chapter III) and Integrated DBOS-Inventory policies.

As it can be seen from the figure, both policies are equivalent in performance. A look at the inventory variables (a snapshot is shown in Figure 4.5) shows clearly the intuitive behavior of maintaining the inventories clear all the time. With no lead time, and the purchase price is fixed, there is no reason for stocking and hence inventories are empty. This confirms the fact that when inventory motivation is absent, Integrated DBOS-Inventory policy behaves exactly as DBOS policy.

4.4.2 Case Study VIII

This case study investigates the Integrated DBOS-Inventory policy when a special introductory price of spare batteries is given to the customer fleet company (e.g., part

Table 4.1: Case Study VII Parameters

Parameter	Symbol	Value
Number of vehicles	n	2
Number of loading profiles	m	2
Plan horizon (years)	K	5
Vehicles allocated per loading profile	d_j	[1,1]
Degradation rates (per month)	r_j	[0.04,0.03]
Swapping costs ($)	$c_1(k)$	800
DBOS (only) substitution costs ($)	$c_2(k)$	11600
Integrated DBOS-Inventory Substitution Cost ($)	$c_{sub}(k)$	1600
Purchase Cost ($)	$c_p(k)$	10000
Order Cost ($)	$c_o(k)$	250
Holding Cost ($)	$c_h(k)$	500
Threshold	β	0.1
Discritization Resolution	ϕ	0.01
Deterioration rate of the batteries in inventory	ρ	0.01
ϵ Resolution		3
Horizon Discretization interval (year)	Δ	1
Hidden cost (penalty) coefficient used in policy build-up* ($)	C_{hid}	32000
Proportional penalty of over-usage of batteries used in policy validation ($/0.01 of over-usage in health state)		3000

* This number will be multiplied by a small probability as seen from Equation (3.13)

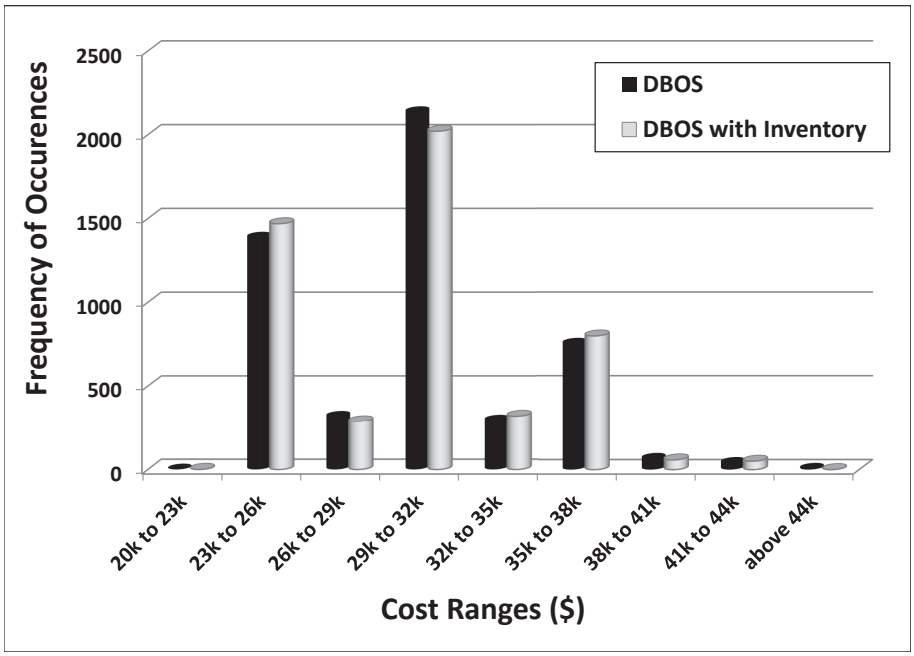


Figure 4.4: Case Study VII (Fixed c_p) Results

of the fleet hybridization or electrification contract). We maintain zero lead time and offer spare batteries at the beginning of the plan horizon at a discounted rate of 25% of their nominal price. While this percentage seems drastic, it is tailored for simulation purposes. It was found out that with shortened plan horizon, inventory deterioration rate, costs of inventory holding, stocking starts to be observed after a reduction of nearly 50% of the nominal battery price. The stocking at this level of reduction is still of one battery. Therefore, the maintenance costs of both stochastic DBOS and Integrated DBOS-Inventory policies are very close and comparisons are hard to establish. The closeness is attributed to the fact that the savings associated with one battery initial stocking is overshadowed by inventory holding costs and inventory deterioration. A reduction of 75% in the original price will initiate stocking of two batteries at the beginning which will enable clearer view of the savings in this small case study (two vehicles), and thus addresses its aim. Hence, the discounted price of 25% is chosen for this case study. In real life scenario, fleets will be comprised of tens

	A	B	C	D	E	F	G
1	Run 1	Initial State 1					
2	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
3	0.03	0.02	0	0	0	0	0
4	0.07	0.04	0	0	0	0	0
5	0.02	0.08	0	0	0	0	0
6	0.06	0.03	0	0	0	0	0
7	0.09	0.07	0	0	0	0	0
8	Run 1	Initial State 2					
9	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
10	0.04	0.05	0	0	0	0	0
11	0.06	0.09	0	0	0	0	0
12	0.1	0.03	0	0	0	0	0
13	0.03	0.06	0	0	0	0	0
14	0.07	0.1	0	0	0	0	0
15	Run 2	Initial State 1					
16	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
17	0.03	0.03	0	0	0	0	0
18	0.08	0.07	0	0	0	0	0
19	0.04	0.02	0	0	0	0	0
20	0.07	0.05	0	0	0	0	0
21	0.1	0.08	0	0	0	0	0
22	Run 2	Initial State 2					
23	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
24	0.04	0.04	0	0	0	0	0
25	0.06	0.07	0	0	0	0	0
26	0.1	0.04	0	0	0	0	0
27	0.05	0.06	0	0	0	0	0
28	0.09	0.09	0	0	0	0	0
29	Run 3	Initial State 1					
30	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
31	0.04	0.04	0	0	0	0	0
32	0.07	0.07	0	0	0	0	0
33	0.05	0.03	0	0	0	0	0
34	0.03	0.06	0	0	0	0	0
35	0.06	0.1	0	0	0	0	0
36	Run 3	Initial State 2					

Figure 4.5: Snapshot of Health State and Inventory Variables from Some of the Runs in Case Study VII

or hundreds of vehicles where stocking will be in larger numbers and savings can be more discernible.

We compare maintenance plan costs associated with the Integrated DBOS-Inventory policy with the ones associated with standard stochastic DBOS policy. Figure 4.6 shows the results. It can be seen clearly that the costs are higher with stochastic DBOS. With no inventory, this policy cannot capitalize on special introductory price. Figure 4.7 shows a snapshot of the inventory variables in a number of the runs. As it can be seen, the policy stocks two spare batteries each time initially. Nonetheless, what the policy does with these stocking is different from run to run depending on the stochastic nature of the health state evolution. For example we show here 3 different scenarios (marked by colors of the rectangles shown in the figure) where the

two battery inventory has been managed differently each time.

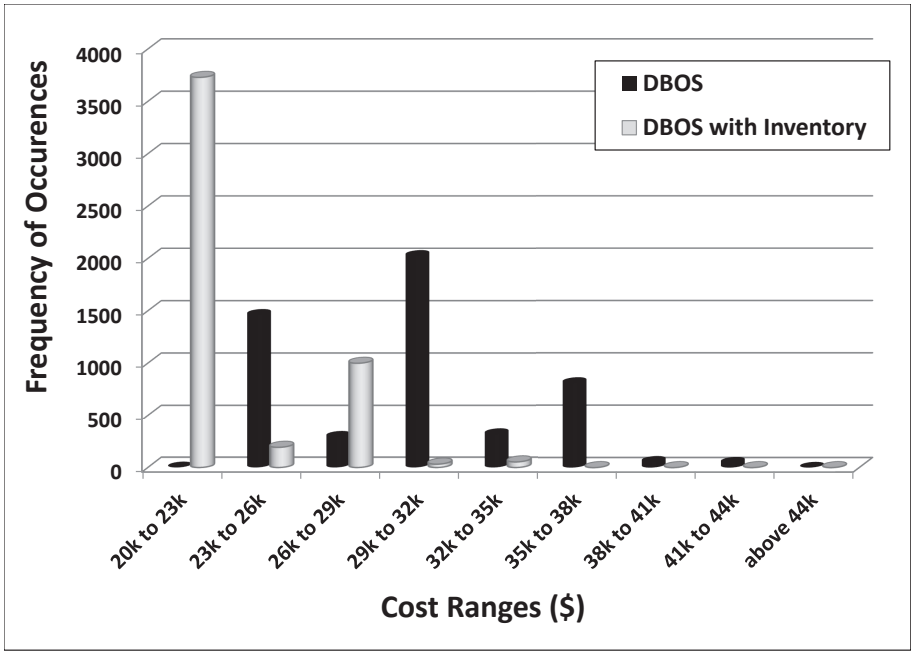
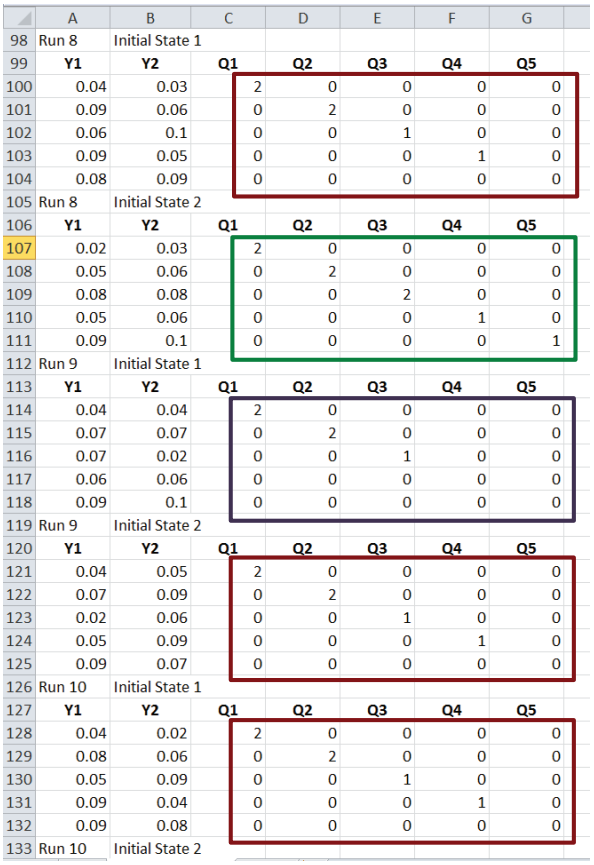


Figure 4.6: Case Study VIII (Special Introductory Price) Results

The conclusion of this case study is that Integrated DBOS-Inventory policy takes the action of stocking spare batteries when the opportunity of savings allows for that. While inventory deterioration and inventory holding costs might reduce savings, stocking batteries can still provide improvement over stochastic DBOS policy.

4.4.3 Case Study IX

The third case study in this Chapter aims to investigate the Integrated DBOS-Inventory policy behavior when the battery prices are increasing with time (ramp-up price). This situation might not be directly intuitive. On the one hand, Li-ion batteries, as many other technologies, are undergoing significant development where new models with cheaper production prices are emerging. On the other hand, the Li-ion newer generations might not be compatible with the fleet who has undergone hybridization or electrification many years and thus many system generations ago.



	A	B	C	D	E	F	G
98	Run 8	Initial State 1					
99	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
100	0.04	0.03	2	0	0	0	0
101	0.09	0.06	0	2	0	0	0
102	0.06	0.1	0	0	1	0	0
103	0.09	0.05	0	0	0	1	0
104	0.08	0.09	0	0	0	0	0
105	Run 8	Initial State 2					
106	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
107	0.02	0.03	2	0	0	0	0
108	0.05	0.06	0	2	0	0	0
109	0.08	0.08	0	0	2	0	0
110	0.05	0.06	0	0	0	1	0
111	0.09	0.1	0	0	0	0	1
112	Run 9	Initial State 1					
113	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
114	0.04	0.04	2	0	0	0	0
115	0.07	0.07	0	2	0	0	0
116	0.07	0.02	0	0	1	0	0
117	0.06	0.06	0	0	0	0	0
118	0.09	0.1	0	0	0	0	0
119	Run 9	Initial State 2					
120	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
121	0.04	0.05	2	0	0	0	0
122	0.07	0.09	0	2	0	0	0
123	0.02	0.06	0	0	1	0	0
124	0.05	0.09	0	0	0	1	0
125	0.09	0.07	0	0	0	0	0
126	Run 10	Initial State 1					
127	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
128	0.04	0.02	2	0	0	0	0
129	0.08	0.06	0	2	0	0	0
130	0.05	0.09	0	0	1	0	0
131	0.09	0.04	0	0	0	1	0
132	0.09	0.08	0	0	0	0	0
133	Run 10	Initial State 2					

Figure 4.7: Snapshot of Health State and Inventory Variables from Some of the Runs in Case Study VIII

The compatible old-generation batteries might then undergo price increases as less manufacturing facilities and production are involved with these generations. All of this is driven by the supply and demand. A living example of this situation can be seen in the market and prices of the internal memory used in personal computers and Laptops, known as Random Access Memory (RAM). With the emergence of the faster and newer DDR3 RAM, the DDR2 RAM which is needed for many older motherboards has undergone significant price increases. Currently, DDR3 RAM can be acquired as much as half the price of DDR2, despite the latter being slower. Similar thing happened when DDR2 RAM replaced DDR RAM several years ago. Therefore, while technological products seem to undergo price decreases generation after another, the compatibility might change this point of view to price increases. Finally, the price

increases can as well be justified by inflation as the prices of materials and energy used in the production of the batteries are increasing.

We increase the price of acquiring new batteries each interval by \$1600, where the order cost is maintained. With this increase, the hidden cost (penalty) coefficient used in policy build-up (c_{hid}) has to be increased as well. When the batteries are becoming expensive, the policy will opt for over-usage if the penalty is no longer proportional to the battery price, and we would see higher percentages of over-usages. The proportional penalty of over-usage of batteries used in policy validation does not have to be changed as it is merely used after the policy has been formulated and do not affect the percentages of over-usages. We maintain the same penalties for both policies in the validation part for fair comparison. Figure 4.8 shows the results. It is clearly that the Integrated DBOS-Inventory policy not only has lower maintenance plan costs in general, but also shows robust behavior in maintaining over 70% of the runs within the same cost category. The policy capitalizes on being well informed of the future price increases. Hence it takes advantage of the lower prices at the beginning, specifically the first two intervals where it stocks batteries for later use when these batteries are more expensive to purchase. Figure 4.9 shows a snapshot of the inventory variables. Once again, it can be seen that different runs will have different stocking and inventory evolutions (marked in the figure by the color of the frame box). The stochastic nature of the health state evolution governs that.

We note that there is less than 10% of the runs where the stochastic DBOS policy had low costs. A detailed look into the health state evolution in these runs revealed that the predicted degradation was lower than expected in several of the intervals, allowing the policy to avoid additional expensive substitutions at later stages.

The conclusion of this case study is that Integrated DBOS-Inventory policy can benefit from being informed about future prices by taking the action of stocking spare batteries. While inventory deterioration and inventory holding costs might reduce

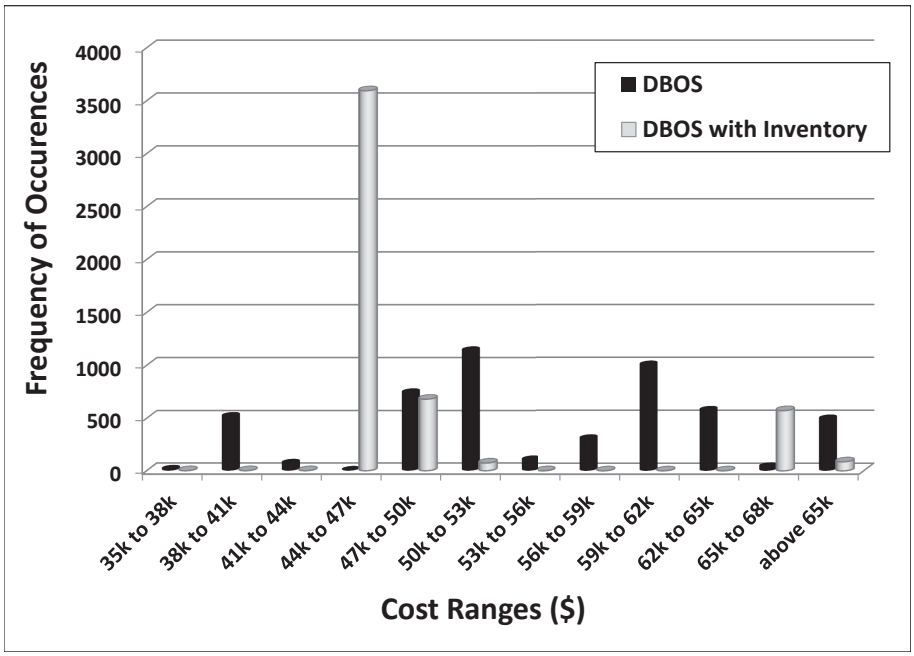
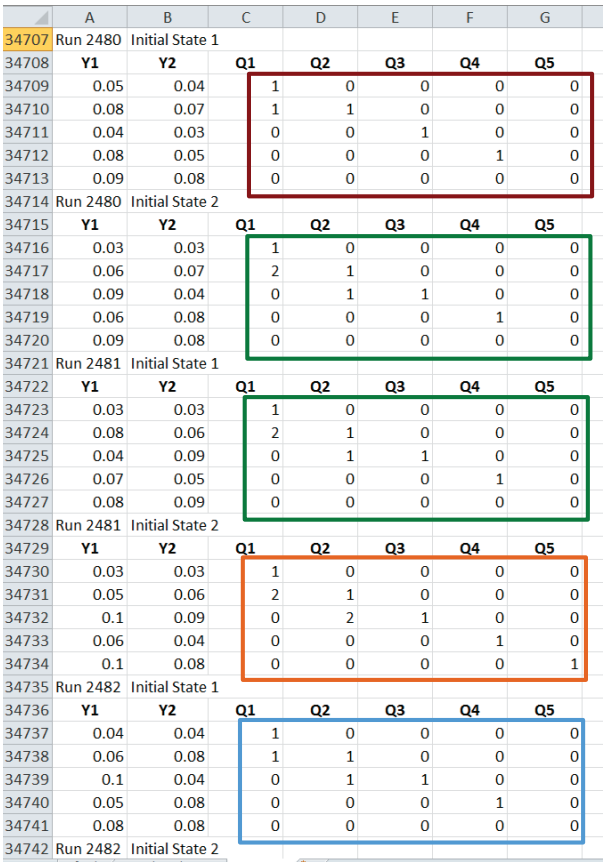


Figure 4.8: Case Study IX (Ramp up Price) Results

savings, stocking batteries can still provide improvement over stochastic DBOS policy.

4.4.4 Case Study X

The final case study aims to investigate the primary reason for acquiring an inventory; a non-zero lead time. We apply the Integrated DBOS-Inventory policy on a case study with similar parameters to the ones in Case Study VII, but with lead time (λ) equals to one and two intervals. We allow for rush replenishment (from local supplier or competitor) with high costs equivalent to \$20k per battery. This is a next state cost as explained in Section 4.3.2.4. The policy will trade off excessively overusing the batteries and ordering this rush expensive replinishment when things do not work as well as the policy has planned for. In previous cases, rush replinshment made no value as the policy can substitute batteries (with the normal cost of purchase) when overusing is imminent. In this case, ordering a new battery will take one or two intervals (based on the lead time) which requires extensive planning.



	A	B	C	D	E	F	G
34707	Run 2480	Initial State 1					
34708	Y1	Y2	Q1	Q2	Q3	Q4	Q5
34709	0.05	0.04	1	0	0	0	0
34710	0.08	0.07	1	1	0	0	0
34711	0.04	0.03	0	0	1	0	0
34712	0.08	0.05	0	0	0	1	0
34713	0.09	0.08	0	0	0	0	0
34714	Run 2480	Initial State 2					
34715	Y1	Y2	Q1	Q2	Q3	Q4	Q5
34716	0.03	0.03	1	0	0	0	0
34717	0.06	0.07	2	1	0	0	0
34718	0.09	0.04	0	1	1	0	0
34719	0.06	0.08	0	0	0	1	0
34720	0.09	0.08	0	0	0	0	0
34721	Run 2481	Initial State 1					
34722	Y1	Y2	Q1	Q2	Q3	Q4	Q5
34723	0.03	0.03	1	0	0	0	0
34724	0.08	0.06	2	1	0	0	0
34725	0.04	0.09	0	1	1	0	0
34726	0.07	0.05	0	0	0	1	0
34727	0.08	0.09	0	0	0	0	0
34728	Run 2481	Initial State 2					
34729	Y1	Y2	Q1	Q2	Q3	Q4	Q5
34730	0.03	0.03	1	0	0	0	0
34731	0.05	0.06	2	1	0	0	0
34732	0.1	0.09	0	2	1	0	0
34733	0.06	0.04	0	0	0	1	0
34734	0.1	0.08	0	0	0	0	1
34735	Run 2482	Initial State 1					
34736	Y1	Y2	Q1	Q2	Q3	Q4	Q5
34737	0.04	0.04	1	0	0	0	0
34738	0.06	0.08	1	1	0	0	0
34739	0.1	0.04	0	1	1	0	0
34740	0.05	0.08	0	0	0	1	0
34741	0.08	0.08	0	0	0	0	0
34742	Run 2482	Initial State 2					

Figure 4.9: Snapshot of Health State and Inventory Variables from Some of the Runs in Case Study IX

One important note we would like to make about this case study is the fact that with lead time, we can only apply standard stochastic DBOS if all orders are rush (and expensive) orders. Figure 4.10 shows the costs distribution of the different runs for the DBOS with rush orders policy, and the Integrated DBOS-Inventory policy with lead time equal to one and two intervals.

Intuitively, the figure shows that the rush expensive orders force DBOS without inventory to sustain substantial maintenance costs which allows the proposed integrated policy to significantly outperform that by a large margin. More importantly, comparing the Integrated DBOS-Inventory policy results for ($\lambda = 1$) and ($\lambda = 2$) reveals the smartness of the proposed policy in handling different lead times while maintaining relatively close performances. It is undoubted that with increased lead

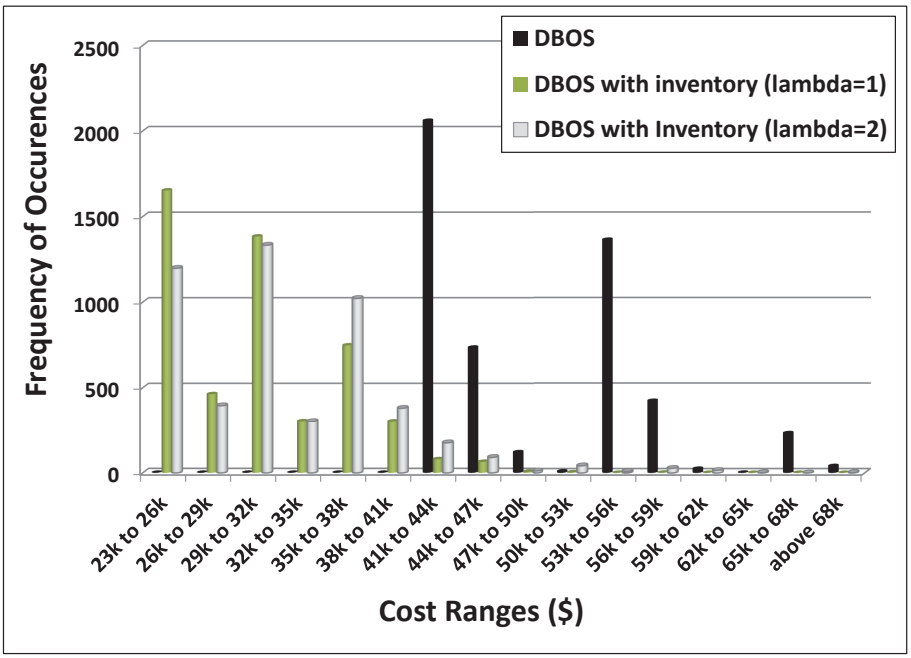


Figure 4.10: DBOS Policy with Rush Orders and Integrated DBOS Inventory Policy (with $\lambda = 1$ *and* 2) Results

time, any predictive policy will suffer a decrease in the performance, and this is what we note for the lower cost category occurrences. However, it can be seen that the loss is significantly small (few hundred occurrences in the lowest cost range only, and maintaining equivalent occurrences in other ranges). This reflects the capability of the proposed policy in adjusting to lead time and making predictive orders accordingly.

Studying the evolution of the Q_q and $Q_{pending}$ variables (the latter only appears when $\lambda = 2$) in Figures 4.11 and 4.12 reveals this predictive behavior. One or two intervals (depending on λ) prior to the point at which substitution is taking place, a purchase order is made. In most of the outcomes, the substitution will take place directly upon the arrival of the ordered batteries (the action that follow the moment when the orders are in Q_1). This prevents actual inventory stocking (Q_q for $q \geq 2$ are zeros), and hence avoiding inventory holding costs (similar to the rational in Case Study VII). However, due to the stochastic nature of the health state evolution, there will be a limited number of occurrences when the ordered battery or batteries

are actually stocked for one interval. This happens when the policy would have anticipated substitution in the next interval; however the degradation of the batteries in the field turned out to be "nicer than expected" and was smaller. There is an incredible outcome of such behavior. As the proposed policy with fixed prices (even when there is lead time) avoids any long-term stocking, we can heuristically ignore modeling older batteries inventory levels (Q_q for large q), which reduces the state-action space dimension. What exact value of q after which we can ignore modeling the inventory level is dependent on the degradation rates, degradation prediction errors and their probability distribution. It can be selected upon running small size fleets as in our case here. The case study provided in Appendix B will benefit from this heuristic tackling of the policy's curse of dimensionality.

We note also that there is a number of rush replenishment (less than 8% when $\lambda = 1$, and 23% when $\lambda = 2$). Intuitively, with larger lead time more rush replenishment are required due to the increased steps ahead prediction of the health states. These rush replenishment are responsible for the high costs (above \$38k). This is the outcome of the current setup of our problem, having the penalty for the potential of overusage during policy buildup, penalty for actual overusage for policy validation, and penalty for rush replenishment (see Sections 3.2.5 and 4.3.2.4). Changing these parameters will yield different numbers. For example, since all rush replenishment are occurring in the final interval (which confirms that they are caused by prediction malfunction), we can reduce the penalty of the potential of overusing in the last intervals to allow the policy a more bolder attitude where it goes for overusage rather than rush replenishment.

We benchmark the cost results with a more intelligent policy than DBOS with rush orders. Referring back to the standard replenishment inventory policies, briefly mentioned in Section 4.1, we can combine a version of the standard periodic review (R,S) inventory policy with DBOS framework to generate a policy that is guaranteed

	A	B	C	D	E	F	G	H	I
103	Run 9	Initial State 2							
104	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Rush Rplnshmnt**	
105	0.02	0.05	0	0	0	0	0	0	
106	0.05	0.1	1	0	0	0	0	0	
107	0.09	0.04	1	0	0	0	0	0	
108	0.05	0.08	0	0	0	0	0	0	
109	0.09	0.05	0	0	0	0	0	1	
110	Run 10	Initial State 1							
111	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Rush Rplnshmnt**	
112	0.04	0.03	0	0	0	0	0	0	
113	0.07	0.06	1	0	0	0	0	0	
114	0.03	0.09	1	0	0	0	0	0	
115	0.06	0.05	0	0	0	0	0	0	
116	0.09	0.09	0	0	0	0	0	0	
117	Run 10	Initial State 2							
118	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Rush Rplnshmnt**	
119	0.03	0.05	0	0	0	0	0	0	
120	0.06	0.08	1	0	0	0	0	0	
121	0.09	0.05	1	0	0	0	0	0	
122	0.03	0.08	1	0	0	0	0	0	
123	0.07	0.04	0	0	0	0	0	0	
124	Run 11	Initial State 1							
125	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Rush Rplnshmnt**	
126	0.05	0.04	0	0	0	0	0	0	
127	0.09	0.06	2	0	0	0	0	0	
128	0.03	0.09	0	1	0	0	0	0	
129	0.06	0.04	0	0	0	0	0	0	
130	0.08	0.09	0	0	0	0	0	0	
131	Run 11	Initial State 2							
132	**Y1**	**Y2**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Rush Rplnshmnt**	
133	0.03	0.03	0	0	0	0	0	0	
134	0.06	0.08	1	0	0	0	0	0	
135	0.08	0.03	1	0	0	0	0	0	
136	0.04	0.07	0	0	0	0	0	0	
137	0.08	0.1	0	0	0	0	0	0	
138	Run 11	Initial State 1							

Figure 4.11: Snapshot of Health State and Inventory Variables from Some of the Runs of the Proposed Policy When $\lambda = 1$

to be optimal on the fleet management side (due to the merits of DBOS). This way we can compare the optimality of the inventory replenishment side between the Integrated DBOS-Inventory policy and the combined DBOS-(R,S) policy. Clearly, the periodic interval at which the inventory is reviewed is the same as the discretization interval (Δ). We choose the upper limit (S) to equal 2 batteries as we have found that this shows the best performance in terms of costs. Results comparing our proposed policy, DBOS with rush orders policy, and the combined DBOS-(R,S) policy for lead time equal to one are shown in Figure 4.13. From the figure, we can see that combining DBOS and the (R,S) policy outperforms DBOS with rush orders. Additionally, we note that with the combined policy, no rush orders has been recorded, confirming

	A	B	C	D	E	F	G	H	I
163	Run 14	Initial State 2							
164	**Y1**	**Y2**	**$Q_{pending}1$**	**Q1**	**Q2**	**Q3**	**Q4**	**Rush RpInshmnt**	
165	0.03	0.05	2	0	0	0	0	0	
166	0.07	0.09	0	2	0	0	0	0	
167	0.03	0.04	1	0	0	0	0	0	
168	0.07	0.08	0	1	0	0	0	0	
169	0.11	0.04	0	0	0	0	0	0	
170	Run 15	Initial State 1							
171	**Y1**	**Y2**	**$Q_{pending}1$**	**Q1**	**Q2**	**Q3**	**Q4**	**Rush RpInshmnt**	
172	0.04	0.02	2	0	0	0	0	0	
173	0.07	0.04	0	2	0	0	0	0	
174	0.03	0.09	0	0	1	0	0	0	
175	0.05	0.04	0	0	0	0	0	0	
176	0.08	0.08	0	0	0	0	0	0	
177	Run 15	Initial State 2							
178	**Y1**	**Y2**	**$Q_{pending}1$**	**Q1**	**Q2**	**Q3**	**Q4**	**Rush RpInshmnt**	
179	0.03	0.03	2	0	0	0	0	0	
180	0.06	0.05	0	2	0	0	0	0	
181	0.1	0.09	0	0	2	0	0	0	
182	0.05	0.06	0	0	0	0	0	0	
183	0.1	0.1	0	0	0	0	0	0	
184	Run 16	Initial State 1							
185	**Y1**	**Y2**	**$Q_{pending}1$**	**Q1**	**Q2**	**Q3**	**Q4**	**Rush RpInshmnt**	
186	0.04	0.04	0	0	0	0	0	0	
187	0.07	0.07	2	0	0	0	0	0	
188	0.04	0.03	0	0	0	0	0	0	
189	0.08	0.06	1	0	0	0	0	0	
190	0.05	0.09	0	0	0	0	0	0	
191	Run 16	Initial State 2							
192	**Y1**	**Y2**	**$Q_{pending}1$**	**Q1**	**Q2**	**Q3**	**Q4**	**Rush RpInshmnt**	
193	0.02	0.04	2	0	0	0	0	0	
194	0.05	0.09	0	2	0	0	0	0	
195	0.09	0.03	0	0	1	0	0	0	
196	0.04	0.05	0	0	0	0	0	0	
197	0.07	0.08	0	0	0	0	0	0	
198	Run 17	Initial State 1							

Figure 4.12: Snapshot of Health State and Inventory Variables from Some of the Runs of the Proposed Policy When $\lambda = 2$

the significant performance and proving the intelligence of the combined policy over rush orders. Now comparing the combined policy with our proposed policy, it can be clearly noticed that our proposed Integrated DBOS-Inventory policy performance is certainly unmatched. This confirms the optimality of the inventory replenishment in our proposed policy since it's being looked at in a centralized manner in comparison to the decentralization occurring when combining DBOS with (R,S) replenishment policy.

We attempt to benchmark the cost results from another point of view. Integrated DBOS-Inventory policy represents an advanced prognostics and health management (PHM) decision support tool as it connects PHM to inventory management. As

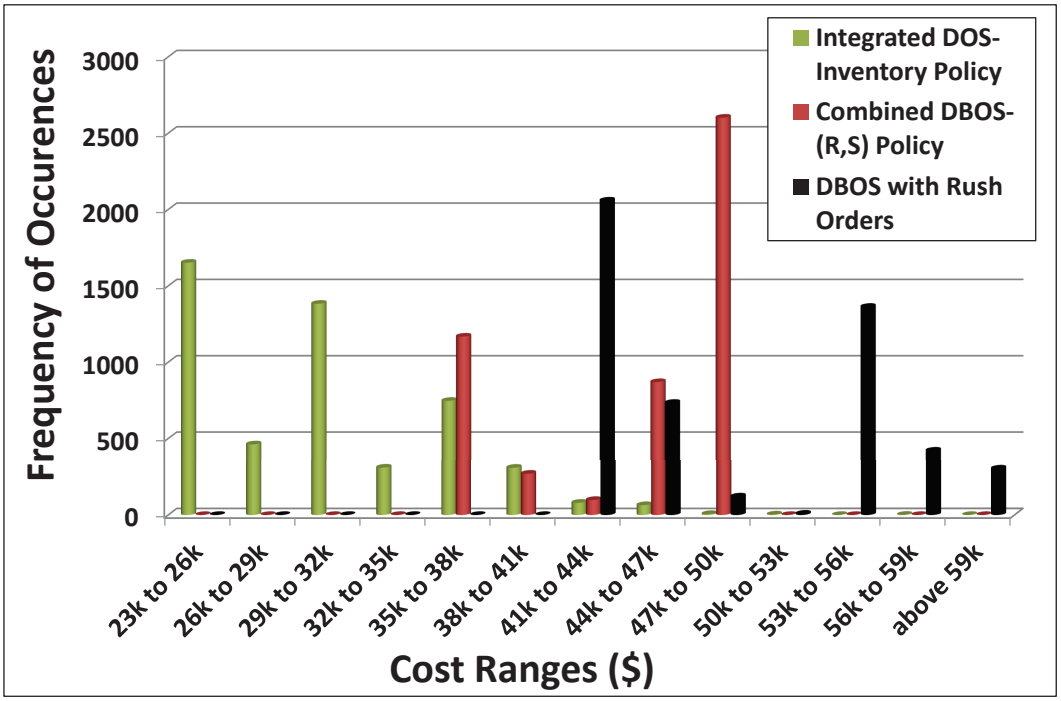


Figure 4.13: Benchmarking the Proposed Policy with Standard Inventory Replenishment Policy

we had mentioned briefly in Chapter I, maintenance paradigm has evolved from the fail and fix approach (reactive maintenance), to periodic maintenance (preventive maintenance), and then to the condition-based maintenance (CBM), finally arriving at PHM. In the basic approach of CBM, the health state change is monitored and upon crossing a predetermined threshold, a maintenance action is triggered. PHM represents an evolution of that due to the incorporation of the prediction of the health state change in PHM, enabling predictive maintenance as well as predictive operations management. When MLDT (see Section 4.1) is significant, the importance of connecting the maintenance with the supply chain logistics has been highlighted by several researchers. Parlier [125] has proposed connecting CBM with supply chain management for US military applications. In [126], he presented the significance of applying such scheme on the maintenance of the AH-64 Apache nose gearbox with almost half a million dollars in savings. We will similarly use the concept of connecting

CBM with inventory management to benchmark our proposed policy.

We use the health state degradation to form a DBOS policy that will purchase batteries when the accumulative degradation in the health states of the batteries exceeds certain limit (threshold). The results for lead time that equals one interval and different thresholds are shown in Figure 4.14. When comparing these results to our proposed policy results for a lead time equal to one (Figure 4.10 or Figure 4.13), the CBM-based policy is outperformed by a significant margin. The proposed policy worst occurrences are in the same cost category as the best occurrences of the CBM-based policy (which prevented a clear plotting of them together on the same graph). While this concludes the benchmarking objective, we point out that this poor performance of the CBM-based policy is certainly counter-intuitive. The policy fails even to outperform the DBOS with rush orders. The primary reason behind this is the accelerated degradation with shortened plan horizon chosen in the case study parameters. Further explanations of this are beyond the scope of the main part of the thesis and will be detailed in Appendix B. We also report there another case study with more "relaxed" degradation and with longer horizon, where the CBM-based policy outperforms DBOS with rush orders.

The conclusion of this case study is that Integrated DBOS-Inventory policy is the best policy that can handle problems with lead time. The policy practices pre-ordering λ intervals before the actual substitution takes place in the larger portion of occurrences. The policy with lead time and fixed prices will opt for empty inventories as much as possible to reduce inventory associated costs. Hence, the policy achieves a combination of optimal inventory replenishment and optimal fleet management.

4.5 Conclusion

Inventory management spells the difference between corporate success and failure. With respect to the fleet management plans, a local inventory inclusion is necessary

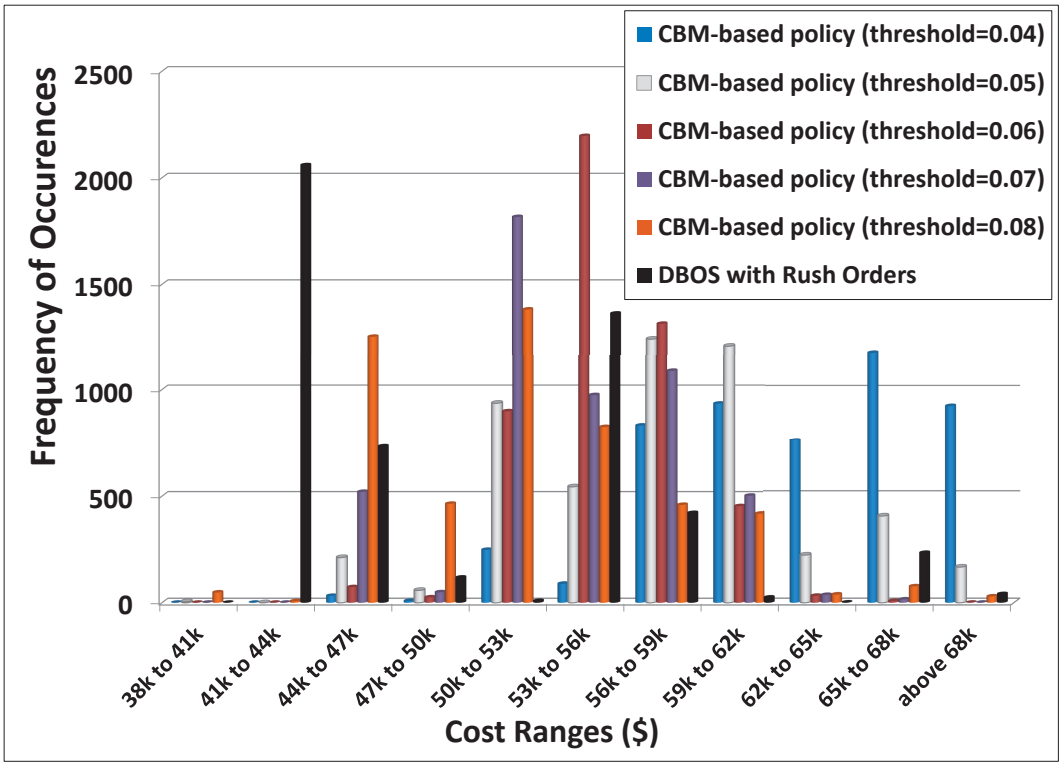


Figure 4.14: Benchmarking the Proposed Policy with CBM-based Policy

when there is lead time. This inclusion is also preferable when special pricing circumstances such as the increase in the cost of the replacing components or the availability of a special introductory price for the spare parts in the original contract. With respect to fleet level battery utilization, the inclusion of inventory was challenging due to the phenomenon of inventory deterioration. We presented in this Chapter a new policy, denoted as the Integrated DBOS-Inventory policy, where deteriorating inventory management was included to the DBOS policy developed in previous chapters. The policy combines the merits of both, achieving optimal placements, optimal substitutions, and optimal replenishing. Through several case studies, the policy was shown to be successful in capturing the rational of the motivation of inventory. Avoidance of any unnecessary stocking in inventory was noticed in Case Studies VII and X. Case Studies VIII and IX showed the capitalization of the policy on inventory to achieve savings in special pricing circumstances compared to the Stochastic DBOS policy.

CHAPTER V

Conclusions and Contributions

5.1 Conclusions

In this dissertation, we proposed a uniquely formulated degradation-informed resource allocation policy, which can provide enhanced utilization for systems of identical assets or components which are used differently. The policy, denoted as Degradation-based Optimal Swapping (DBOS) policy, promotes this utilization through a series of optimally chosen swapping and reset (substitution) actions. The policy takes advantage of the different degradation rates of the components within the system, based on loading conditions or frequency of use, to choose optimal placements of these components. The policy's proven enhanced utilization promotes it to be a key decision support tool in reference to maintenance management where its generic feature allows its application in numerous disciplines. The primary application in this thesis which is the fleet-level battery utilization is motivated by the ongoing plans for electrification and hybridization of ground vehicle fleets. The plans aim to minimize the overall cost of operation and fuel consumption and adhere to environmentally friendly awareness. However, the hybridization projects increased maintenance costs especially for highly costly and degradable components such as Li-ion batteries. Therefore, DBOS can provide an enhanced utilization decreasing the projected maintenance costs significantly. The development of the policy in this primary application

was conducted in three stages:

First, a high confidence (deterministic) degradation estimation has been adopted to uniquely formulate the degradation-informed resource allocation planning scheme. The optimization of the generated mathematical model which represented a (ZOINLP) problem has been investigated. A DBOS-policy-specific algorithm has been developed and successfully implemented. Numerical results showed the strong performance of the algorithm in comparison to standard optimization techniques. Numerical results validated the role of the discretization interval in the DBOS policy, allowing but not necessary choosing the option to perform additional swapping actions minimizing the costly substitution ones. Finally, DBOS was benchmarked with other fleet management maintenance plans where DBOS has been shown to significantly outperform them.

Second, we extended the problem to include uncertain degradation estimation. The formulation of the Stochastic DBOS policy based on the framework of stochastic dynamic programming and Markov Decision Processes has been achieved through modifications to the deterministic model. The modifications have generated several complications, which have been comprehensively solved for small fleets problems. Numerous case studies have shown Stochastic DBOS to outperform Deterministic DBOS. Stochastic DBOS has been tested against variation in uncertainty, lack of knowledge of uncertainty distribution, and lower resolution of error. Results have shown Stochastic DBOS's ability to avoid "bad luck" runs robustly, and hence avoid excessive maintenance costs. Stochastic DBOS has also outperformed Deterministic DBOS in the cases where wrong distributions were used for the policy build-up.

Third, we further extended the problem to include a local inventory management for the spare components involved with the substitution actions. With respect to the fleet management plans, a local inventory inclusion is necessary when there is lead time. This inclusion is also preferable when special pricing circumstances such as the

increase in the cost of the replacing components or the availability of a special introductory price for the spare parts in the original contract. With respect to fleet level battery utilization, the inclusion of inventory was challenging due to the phenomenon of inventory deterioration. A new policy, denoted as the Integrated DBOS-Inventory policy, was proposed, modeled, and tested. Perishable inventory management was integrated to the DBOS policy developed in previous chapters. The policy combines the merits of both, achieving optimal placements, optimal substitutions, and optimal replenishing. Through several case studies, the policy was shown to be successful in capturing the rational behind the motivation of inventory. Avoidance of any unnecessary actual stocking in inventory was shown in cases where motivation for inventory was lacking, while capitalization of the policy on inventory to achieve savings was seen in special pricing circumstances when compared to the Stochastic DBOS policy. The policy was benchmarked with other standard inventory replenishment policies, where the proposed policy outperformed them by large margins.

5.2 Contributions

This dissertation has four main contributions. First, we introduced the unprecedented degradation-informed resource allocation principle denoted as Degradation-based Optimal Swapping (DBOS). This principle promotes enhanced utilization of systems with identical components used differently. Second, we modeled the DBOS concept with high confidence (deterministic) degradation in the health states of the components and we developed a DBOS-specific B&B-based optimization algorithm that is capable of producing repeatable global optimal solutions. Third, we extended the problem to account for uncertain degradation; developing the stochastic DBOS policy built using the framework of Stochastic Dynamic Programming and Markov Decision Processes. The policy showed robust decision making and avoided excessive maintenance costs. The results display strong candidacy to adopt the policy in larger

fleets using Approximate Dynamic Programming techniques. Fourth, we extended the problem further more; developing the Integrated DBOS-inventory policy, which accounted for deteriorating inventory. The policy was shown to acquire optimal replenishment policies in addition to the optimal maintenance management associated with DBOS.

5.3 Proposed Future Work

Future work can be conducted in several directions. We had introduced in Section 1.2 several "straightforward" applications that DBOS or some sort of swapping has been shown to provide utilization and savings. Other prospect applications for DBOS are discussed in the Section 5.3.1. Opportunities for Approximate Dynamic Programming (ADP) implementation is discussed in Section 5.3.2. Section 5.3.3 talks about a future direction that includes augmentation of DBOS to account for fleets comprised of mixed internal combustion (IC) engines vehicles and electric (or hybrid electric) vehicles.

5.3.1 Other Prospect Applications for DBOS

5.3.1.1 Operational Control for Electrical and Mechanical Systems in Residential or Commercial Complexes

Residential or commercial complexes usually involve several electrical and mechanical systems associated with them. Examples of these include elevators, HVAC units (centralized like Chillers and Air Handling Units (AHU) or decentralized like split HVAC units), water pumps, etc. These complexes are built to serve demands in both peak times and off-peak times. For example, an office building needs to have one commercial elevator for every 45,000 net usable square feet, or the ratio of floors of the building to the amount of commercial elevators must be two to one or two-and-

a-half to one, if more people use the building [127]. Apartment and hotel buildings on the other hand differ slightly in regards to the amount and placement of commercial elevators. For a hotel, a building must have one elevator for every seventy-five rooms or one elevator for a three-floor building, while an apartment building requires one commercial elevator for every ninety units, except in urban areas, where the ratio is one commercial elevator for every sixty units [127]. Similarly, HVAC systems can be comprised of several chillers and air handling units working together.

It can be seen that some of these systems might be redundant during off-peak hours, or during less demanding days. For example, the HVAC systems are designed to meet demand during the hottest days of the year; however these are only few days in the year in most regions. To save energy, many building management systems (BMS) have embedded shut down protocols that shuts down some of these systems during off peak hours. The question, which ones should these BMS shut off today, and which ones tomorrow? and the day after? The answer to this question in most BMS systems is rotational swapping (similar to the vehicle tires rotation every 5000 miles). While rotational swapping provides better utilization than direct approach (No Swapping), as has been shown in Section 2.4.1, it does not provide optimal utilization. We are clearly here presented with a system of identical components used differently (in this case it is the frequency rather than loading profile), and these systems are intended to be operational for a finite time (e.g., operational lifetime of an HVAC system ranges from 15 to 20 years [128]).Therefore, DBOS can be beneficial in this case if the degradation rates form the difference in the frequency of use is attainable somehow. The policy can assign which systems to be shutdown in an optimal manner that provides savings in the maintenance plans costs.

5.3.1.2 HR Management

Human beings are by far the most sensitive assets one could attempt to manage. The complex human psychology can nullify most management policies. The Human Resources (HR) departments are just one way corporations and organizations are attempting to manage these complex resources. In terms of degradation, the psychological stress can be viewed as the wearing out of the human asset. The stress and thus the mental state degradation can reach levels where productivity is significantly reduced. At this point, there would be some sort of a reset action (e.g., vacation, company retreat, etc.). However, unlike other assets, the individuality and independence a human being shows makes the outcome of such degradation highly unpredictable. Different humans handle stress differently and have different tolerances. In some disciplines, the individuality is "dialed down" that any individual of the group is capable of performing any of the assignments that is given to the group almost equally. Examples of these are soldiers, and shift workers of the same level.

Soldiers rotate through a cycle of Training, Mobilization, Deployment, Redeployment, Withdraw and Going on Leave, and cycling back to Training and resetting, The newest model of this cycling adopted in the US army is what is known as Army Force Generation (ARFORGEN) and signed on in 2006 has the cycles summarized in three stages Reset, Train/Ready and Available. In the Reset phase, soldiers will return from deployment, have some down-time to re-connect with their families, and return to their regular training schedule. During the Train/Ready phase, units begin to train more extensively, are eligible for deployment, and begin preparing for a specific overseas mission. Finally, in Available, soldiers are ready for deployment. Once a unit deploys and returns to their home station, the cycle repeats itself [129]. While this system was developed to address the shortages in the ready army personnel during 2003-2006 period, it still mimics the rotational fixed swapping (as rotating tires every 5000 miles) in terms of behavior. In this case, reset phase is equivalent to

substitution. Therefore, if the degradation associated with the stress can be propely modeled and forecasted, there is room for DBOS to promote better utilization. Another HR example that DBOS can help in is the medical staff that rotates between the night shift and the day shift.

5.3.2 Stochastic DBOS Curse of Dimensionality and Prospect Solutions

We presented the augmentation of the DBOS policy to account for uncertainty in Chapter III, in what we denoted as the Stochastic DBOS policy. We also showed through several case studies how robust the policy is in terms of avoiding excessive maintenance costs associated with unexpected increase in the degradation. One problem we noted is that Stochastic DBOS, as many SDP instances, suffers from the curse of dimensionality.

The three curses of dimensionality stated by Powell [81] are:

1. State space: If the state variable $S_t = (S_{t1}, S_{t2}, \cdots, S_{ti}, \cdots, S_{tI})$ has I dimensions, and if S_{ti} can take on L possible values, then we might have up to L^I different states.

2. Outcome space: The random variable $W_t = (W_{t1}, W_{t2}, \cdots, W_{tj}, \cdots, W_{tJ})$ might have J dimensions. If W_{tj} can take on M outcomes, then our outcome space might take on up to M^J outcomes.

3. Action space: The decision vector $x_t = (x_{t1}, x_{t2}, \cdots, x_{tk}, \cdots, x_{tK})$ might have K dimensions. If x_{tk} can take on N outcomes, then we might have up to N^K outcomes.

While this for the first instance might sound discouraging, Approximate Dynamic Modeling (ADP) techniques can be utilized to overcome this and promote scalability. ADP techniques have produced production quality solutions to plan the operations of some of the largest transportation companies in the country. These problems require

state variables with millions of dimensions, with very complex dynamics [81]. As a matter of fact, ADP can produce solutions for some problems that are within 1 percent of optimality in a small fraction of the time required to find the optimal solution using classical techniques.

While ADP techniques usually produce sub-optimal results, the significant performance of the stochastic DBOS policy over deterministic DBOS, which itself has been shown to outperform other fleet management policies, presents enough room for sub-optimal solutions to provide further utilization of fleets. Stochastic DBOS represents an ideal candidate for ADP techniques as the structure allows for that. If we observe the transition probability structure, we find several repetitions. Additionally, the state space is blown up because of the health states which again show repetitions. The state space can be as well decomposed into smaller sizes, where the objective function in that case can be changed to promote utilization in a different manner than savings maintenance costs. For example, the 21 possible values of the batteries health states as in Chapter II, can be divided into 3 partitions ([0, 0.06], [0.07, 0.13], [0.14, 0.20]) allowing smaller and tidier computational efforts. However, if we observe the first and second partitions in this case, the maintenance plan cost no longer promotes utilization as the problem will not experience the motivating substitution action. Rather, the problem will eliminate the swapping actions if the maintenance costs are used as the objective function, which counter to our objective. The solution in this case would be in the development of a special objective function that can promote utilization in these ranges.

Other than ADP techniques, decentralizing the problem into several small systems (fleets) with proper communication can provide scalability as well. In this case there will be two levels of swapping, lower level swapping initiated by typical stochastic DBOS applied within smaller fleets and higher level swapping between the small fleets. While decentralization, as ADP, will not guarantee optimal results, the suboptimal

results can maintain enhanced utilization over other policies due to the significant performance of Stochastic DBOS.

5.3.3 DBOS for Mixed Electric Hybrid (or Electric) and Internal Combustion Engine Vehicles

The final future direction we propose is the development of a DBOS policy for fleets of mixed Hybrid (or electric) and Internal Combustion Engines Vehicles. As most of the fleets are not being hybridized all at once, a DBOS policy that accounts for incremental hybridization(or electrification) can be beneficial for current fleet management.

The policy in this case will be more complex. One added complexity is the result of the different degradation of IC engines vehicles and hybrid or electric vehicles with respect to the loading profiles. Another, is the necessity to incorporate gas or diesel consumption in this case as placing one type of vehicles in certain route might significantly increase the gas consumption. On the one hand, as hybrid vehicles are more appealing in the stop-and-go applications, they might be preferable to the IC engine vehicles operating in downtown area for example. However and as we had previously illustrated the projected degradation in the battery health states with this option can be problematic for the fleet operator. Thus the tradeoff can be seen clearly in this problem, and a retrofitted DBOS could provide answers in this case.

APPENDICES

APPENDIX A

Nonlinearity Growth in the Accumulative Degradation with Time in DBOS

The accumulative degradation constraint formulated by Equation (2.11) is the source of nonlinearity in the DBOS policy model. We show here that not only this constraint exhibits nonlinearity in the multiplication of several decision variables, but also it severely grows nonlinearly with the increase of time t. We start with Equation (2.11):

$$y_{it} = (1 - Z_{it})y_{it-1} + \sum_{j=1}^{m} r_j X_{ij}(k), \qquad \forall t = 2, \cdots, T; \quad \forall i = 1, \cdots, n \tag{A.1}$$

Now we substitute for different times starting with $t = 1$. For $t = 1$, the accumulative degradation constraint is:

$$y_{i1} = (1 - Z_{i1})y_{i0} + \sum_{j=1}^{m} r_j X_{ij}(1) = \sum_{j=1}^{m} r_j X_{ij}(1) \qquad \forall i = 1, \cdots, n \tag{A.2}$$

For $t = 2$, the accumulative degradation constraint is:

$$y_{i2} = (1 - Z_{i2})y_{i1} + \sum_{j=1}^{m} r_j X_{ij}(2), \qquad \forall i = 1, \cdots, n \tag{A.3}$$

We substitute in for y_{i1} from above and we get:

$$y_{i2} = (1 - Z_{i2}) \sum_{j=1}^{m} r_j X_{ij}(1) + \sum_{j=1}^{m} r_j X_{ij}(2), \qquad \forall i = 1, \cdots, n \tag{A.4}$$

For $t = 3$, the accumulative degradation constraint is:

$$y_{i3} = (1 - Z_{i3}) y_{i2} + \sum_{j=1}^{m} r_j X_{ij}(3), \qquad \forall i = 1, \cdots, n \tag{A.5}$$

We substitute for y_{i2} from above and we get:

$$y_{i3} = (1 - Z_{i3}) \left((1 - Z_{i2}) \sum_{j=1}^{m} r_j X_{ij}(1) + \sum_{j=1}^{m} r_j X_{ij}(2) \right) + \sum_{j=1}^{m} r_j X_{ij}(3) \tag{A.6}$$

$$= (1 - Z_{i3})(1 - Z_{i2}) \sum_{j=1}^{m} r_j X_{ij}(1) + (1 - Z_{i3}) \sum_{j=1}^{m} r_j X_{ij}(2) + \sum_{j=1}^{m} r_j X_{ij}(3) \tag{A.7}$$

$$\forall i = 1, \cdots, n$$

For $t = 4$, the accumulative degradation constraint is:

$$y_{i4} = (1 - Z_{i4}) y_{i3} + \sum_{j=1}^{m} r_j X_{ij}(4), \qquad \forall i = 1, \cdots, n \tag{A.8}$$

We substitute for y_{i3} from above and we get:

$$
\begin{aligned}
y_{i4} =&(1-Z_{i4})\left((1-Z_{i3})(1-Z_{i2})\sum_{j=1}^{m} r_j X_{ij}(1) + (1-Z_{i3})\sum_{j=1}^{m} r_j X_{ij}(2) + \sum_{j=1}^{m} r_j X_{ij}(3)\right) \\
&+ \sum_{j=1}^{m} r_j X_{ij}(4) \\
=&(1-Z_{i4})(1-Z_{i3})(1-Z_{i2})\sum_{j=1}^{m} r_j X_{ij}(1) + (1-Z_{i4})(1-Z_{i3})\sum_{j=1}^{m} r_j X_{ij}(2) \\
&+ (1-Z_{i4})\sum_{j=1}^{m} r_j X_{ij}(3) + \sum_{j=1}^{m} r_j X_{ij}(4) \qquad \forall i = 1, \cdots, n
\end{aligned} \tag{A.9}
$$

It can be clearly seen that even for a short period of 4, the constraint grows nonlinearly when future accumulative degradation are estimated. This nonlinear growth can significantly cripple some optimization algorithms that are sensitive to highly nonlinear constraints such as SQP with local search.

APPENDIX B

CBM-based Inventory Policy

Overview

In this appendix, we aim to shed more light on the CBM-based (threshold-triggered) inventory replenishment combined with DBOS framework policy, introduced earlier in Section 4.4.4. The policy's poor behavior when applied to the last case study in Chapter IV will be the main discussion of the first part. In the second part, we will present a case study that will reveal a better performance of such policy.

An In-Depth Look into CBM-based Policy Performance in Case Study X

When the CBM-based policy was applied to Case Study X parameters, the results were significantly poor (see Figure B.1). Not only the poor performance was recorded in comparison to DBOS with rush orders, but also the rational behind the CBM-based policy achieving best results at threshold equivalent to 0.08 is as well counter intuitive. With β set at 0.1, it sounds like ordering batteries at 0.08 is too late and is missing the point of the CBM-based policy. The CBM-based policy aims to make the order early

enough (but not too early) using a degradation threshold value, so that the replacing batteries are available to be used when the substitution is needed to take place. With lead time equal to one interval, that means that the order should be placed at a time when the battery's health state will be capable of handling one more interval degradation. With degradation rates equal to 0.04 and 0.03, it becomes intuitive that a threshold of 0.06 or 0.07 should have generated better results. Investigation into the health states evolution reveals the reason behind such results.

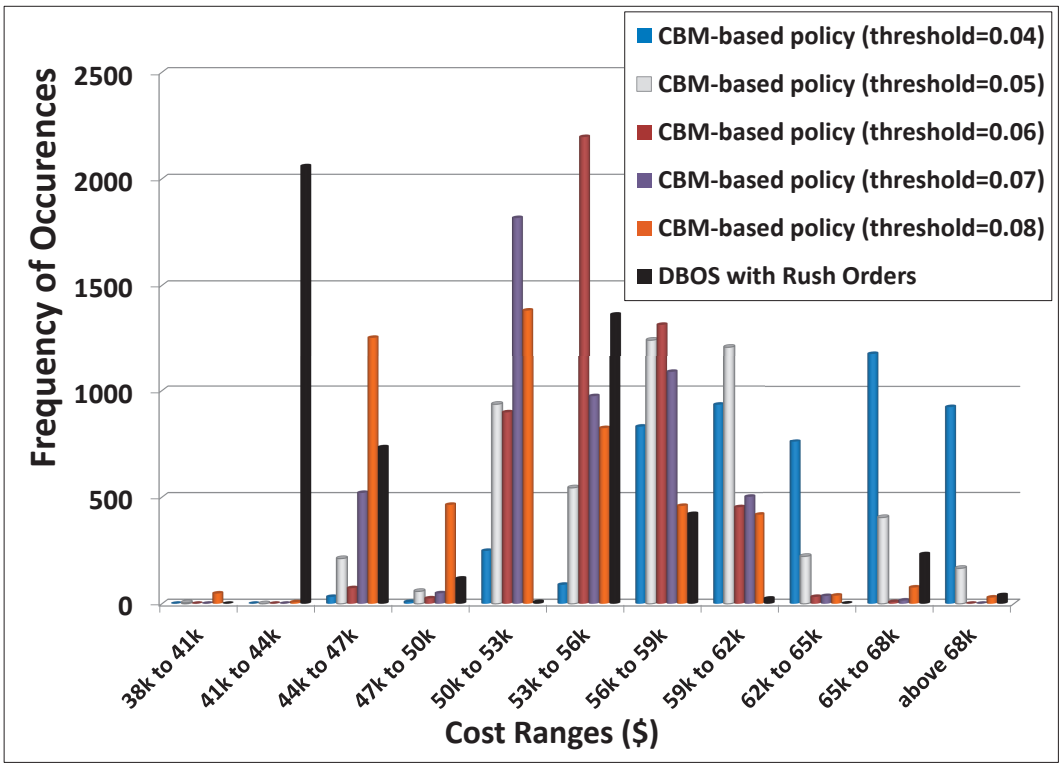


Figure B.1: CBM-based Policy Results When Applied to Case Study X Parameters

The small operational range of health states values and the accelerated degradation (both of which have been selected to accommodate the curse of dimensionality of the policy) are responsible for such malfunction. In specifics, we will talk about three scenarios with selecting the threshold point. First, if the point has been selected to accommodate the degradation rate (e.g., 0.06), then within a range of zero to 0.1, the threshold policy will not have enough time to react and place a strategic purchase

except when the health state is 0.06 exactly. The policy will not react if the health states has not crossed the 0.06 and the policy reaction will be too late if it is after 0.06 (e.g., 0.08) as we will require a substitution before the ordered batteries arrive. The probability of getting exactly 0.06 is significantly small. This means that the policy will make "bad" decisions for all health states except for when it is lucky enough to get the 0.06.

The second scenario occurs when we select the threshold to be small (very early reaction, e.g., 0.04). The problem with such threshold is the continuous and excessive ordering as we are crossing this value so often, which promotes wastefulness. Finally,setting the threshold point at very high value will bring the CBM-based policy to approach the behavior of the DBOS with rush orders. With high threshold, we do not cross that value so often, we therefore are not making enough purchases at non-rush prices, thus we are highly dependent on rush orders to cover the substitutions as the inventory is empty. Therefore, for this case study's parameters, the threshold of 0.08 has shown the "best" results.

In real applications, degradation is not that fast. The CBM-based policy will have enough range to react and this will be shown in the following section.

Case Study XI

To establish the strength of the CBM-based policy, we extend the operational health states range from Case Study X (with $\lambda = 1$) to 0.12, and we reduce the degradation rates to 0.03 and 0.02. We also extend the plan horizon to 9 intervals. Applying the modeling presented in Chapter IV directly will generate a significantly large state action space. The problem will still be in the computational capability of a standard personnel computer. However it will require extensive time. We employ the heuristic fix concerning the role of the older batteries inventory (stock) levels which was developed for fixed price scenarios. We choose (q) at which we ignore the

inventory variables after to be 3. After simulating the problem, we had observed the emptiness of the oldest batteries inventory levels (i.e. all Q_3 values are zero), which confirms our correct q selection as it is an indicator that neither they nor any Q_q with $(q \leq 3)$ play any role in the inventory management.

The results of this case study are shown in Figure B.2. With more relaxed degradation rates and extended health states range, the merits of the CBM-policy are easily recognized. The policy with several threshold values have been able to outperform DBOS with rush orders. The best threshold is 0.09 (which is very close to the performance when the threshold was 0.1 and 0.08), confirming our initial intuition that a threshold selected to accommodate the degradation rates (r here is 0.03 and 0.02) will generate the best CBM-based policy. We also note that when the threshold was selected as high as 0.11, the results are not significant as the rush orders do not represent the best policy anymore. It is thus expected with further relaxation of the degradation and further extension of the health states operational range, that the CBM-based policy will further outperform DBOS with rush orders. Our final remark here is that our proposed Integrated DBOS-Inventory policy is still far from being matched in terms of performance with any of the CBM-based policies.

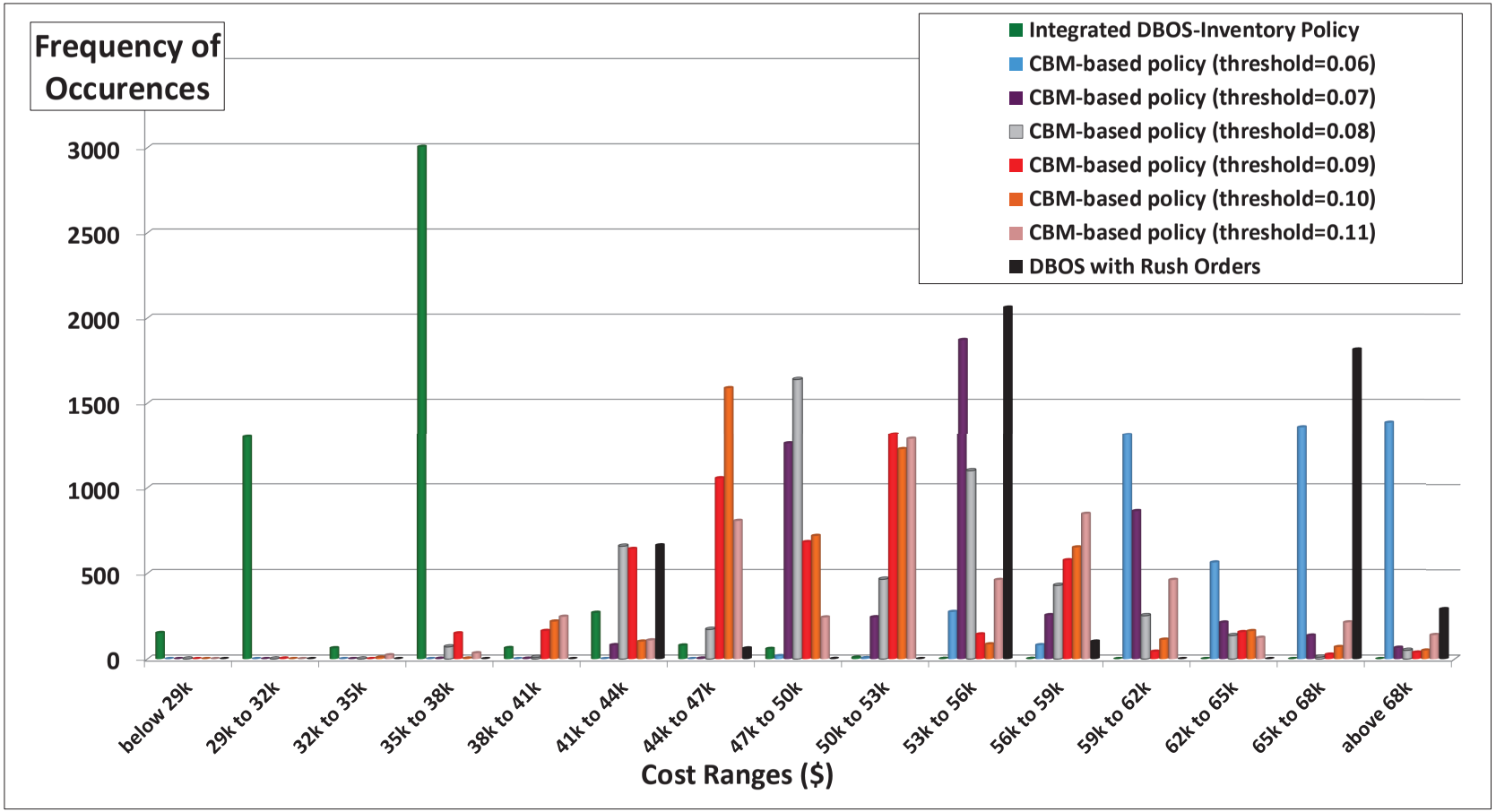


Figure B.2: Integrated DBOS-Inventory Policy, CBM-based Policy, and DBOS with Rush Orders Policy Results When Applied to Case Study XI Parameters

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Koren, *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems*, vol. 75. Wiley, 2010.

[2] R. K. Mobley, *An introduction to predictive maintenance*. Butterworth-Heinemann, 2002.

[3] Metropolitan-Transportation-Authority, "Asset recovery and environmental sustainability," tech. rep., 2013.

[4] L. Gaines, "Review of idling reduction technologies, forward wisconsin reducing diesel emissions for the long haul," tech. rep., US Department of Energy, Argonne National Laboratory, Center for Transportation Research, Chicago, 2005.

[5] USPS, "Postal facts 2013," tech. rep., United States Postal Services, 2013.

[6] GreenBiz.com, "Fedex boosts hybrid delivery truck fleet by 50 percent," *GreenBiz.com*, 2009.

[7] J. Motavalli, "Ups expands its hybrid truck fleet," *The New York Times*, 2010.

[8] J. Schrader, "More cities get on board with hybrid buses," *USA TODAY*, 2008.

[9] M. Maynard, "As hybrid buses get cheaper, cities fill their fleets," *The New York Times*, 2009.

[10] USABC, *Electric Vehicle Battery Test Procedures Manual*. USABC and National Laboratories, 2 ed., 1996.

[11] Frost and Sullivan, "Strategic analysis of north american passenger electric vehicle market," Tech. Rep. N598-18, Frost and Sullivan, 2009.

[12] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007.

[13] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems*. Society for Industrial Mathematics, 2009.

[14] J. Rust, "Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher," *Econometrica*, vol. 55, no. 5, pp. pp. 999–1033, 1987.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 1994.

[16] D. Bertsimas and J. Tsitsiklis, "Introduction to linear optimization," 1997.

[17] A. Almuhtady, S. Lee, E. Romeijn, and J. Ni, "A maintenance-optimal swapping policy for a fleet of electric or hybrid-electric vehicles," in *International Conference on Operations Research and Enterprise Systems*, (Barcelona, Spain), Feb 2013.

[18] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, and G. Sigismondi, "The fleet assignment problem: Solving a large-scale integer program," *Mathematical Programming*, vol. 70, no. 1, pp. 211–232, 1995.

[19] R. Gopalan and K. T. Talluri, "The aircraft maintenance routing problem," *Operations Research*, vol. 46, no. 2, pp. 260–271, 1998.

[20] J. Abara, "Applying integer linear programming to the fleet assignment problem," *Interfaces*, vol. 19, no. 4, pp. pp. 20–28, 1989.

[21] A. Jarrah and G. Yu, *A Model for Airline Maintenance Scheduling.* Working papers, Department of Management Science and Information Systems, University of Texas at Austin, 1990.

[22] L. Clarke, E. Johnson, G. Nemhauser, and Z. Zhu, "The aircraft rotation problem," *Annals of Operations Research*, vol. 69, pp. 33–46, 1997.

[23] K. T. Talluri, "Swapping applications in a daily airline fleet assignment," *Transportation Science*, vol. 30, no. 3, pp. 237–248, 1996.

[24] C. Gao, E. Johnson, and B. Smith, "Integrated airline fleet and crew robust planning," *Transportation Science*, vol. 43, no. 1, pp. 2–16, 2009.

[25] B. Rexing, C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy, "Airline fleet assignment with time windows," *Transportation Science*, vol. 34, no. 1, pp. 1–20, 2000.

[26] M. Lohatepanont and C. Barnhart, "Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment," *Transportation Science*, vol. 38, no. 1, pp. 19–32, 2004.

[27] J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser, "A robust fleet-assignment model with hub isolation and short cycles," *Transportation Science*, vol. 38, no. 3, pp. 357–368, 2004.

[28] B. C. Smith and E. L. Johnson, "Robust airline fleet assignment: Imposing station purity using station decomposition," *Transportation Science*, vol. 40, no. 4, pp. 497–516, 2006.

[29] M. E. Berge and C. A. Hopperstad, "Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms," *Operations Research*, vol. 41, no. 1, pp. 153–168, 1993.

[30] A. I. Jarrah, J. Goodstein, and R. Narasimhan, "An efficient airline re-fleeting model for the incremental modification of planned fleet assignments," *Transportation Science*, vol. 34, no. 4, pp. 349–363, 2000.

[31] L. Gaines, "Truck idling: Implications and solutions, alternatives to truck engine idling," tech. rep., US Department of Energy, Argonne National Laboratory, Center for Transportation Research, Chicago, 2004.

[32] American-Trucking-Associations, "Trucking industry facts," tech. rep., 2010.

[33] American-Trucking-Associations, "Tire pressure monitoring and inflation," *Heavy Duty Trucking Magazine*, 2008.

[34] F. M. C. S. Administration, *Federal Motor Carrier Safety Regulations - 393.75*. U.S. Department of Transportation, Washington, DC, 2005. Part 393, Subpart G - Miscellaneous parts and accessories 393.75, Tires.

[35] A. Almuhtady, S. Lee, and J. Ni, "Degradation-based swapping policy with application to system-level manufacturing utilization," in *International Manufacturing Science and Engineering Conference - MSEC2012*.

[36] A. Almuhtady, S. Lee, and J. Ni, "Joint maintenance and production planning by maintenance-optimal swapping," in *International Manufacturing Science and Engineering Conference - MSEC2013*.

[37] F. Badıa, M. Berrade, and C. A. Campos, "Optimal inspection and preventive maintenance of units with revealed and unrevealed failures," *Reliability Engineering & System Safety*, vol. 78, no. 2, pp. 157–163, 2002.

[38] V. Mijailovic, "Probabilistic method for planning of maintenance activities of substation components," *Electric Power Systems Research*, vol. 64, no. 1, pp. 53–58, 2003.

[39] M. Chen and R. M. Feldman, "Optimal replacement policies with minimal repair and age-dependent costs," *European Journal of Operational Research*, vol. 98, no. 1, pp. 75–84, 1997.

[40] F. Barbera, H. Schneider, and P. Kelle, "A condition based maintenance model with exponential failures and fixed inspection intervals," *Journal of the Operational research Society*, pp. 1037–1045, 1996.

[41] M. Marseguerra, E. Zio, and L. Podofillini, "Condition-based maintenance optimization by means of genetic algorithms and monte carlo simulation," *Reliability Engineering & System Safety*, vol. 77, no. 2, pp. 151–165, 2002.

[42] R. Yam, P. Tse, L. Li, and P. Tu, "Intelligent predictive decision support system for condition-based maintenance," *The International Journal of Advanced Manufacturing Technology*, vol. 17, no. 5, pp. 383–391, 2001.

[43] Z. M. Yang, D. Djurdjanovic, and J. Ni, "Maintenance scheduling in manufacturing systems based on predicted machine degradation," *Journal of Intelligent Manufacturing*, vol. 19, no. 1, pp. 87–98, 2008.

[44] L. HAU and M. J. Rosenblatt, "A production and maintenance planning model with restoration cost dependent on detection delay," *IIE transactions*, vol. 21, no. 4, pp. 368–375, 1989.

[45] K. L. Cheung and W. H. Hausman, "Joint determination of preventive maintenance and safety stocks in an unreliable production environment," *Naval Research Logistics (NRL)*, vol. 44, no. 3, pp. 257–272, 1997.

[46] N. Rezg, X. Xie, and Y. Mati, "Joint optimization of preventive maintenance and inventory control in a production line using simulation," *International Journal of Production Research*, vol. 42, no. 10, pp. 2029–2046, 2004.

[47] X. Yao, X. Xie, M. C. Fu, and S. I. Marcus, "Optimal joint preventive maintenance and production policies," *Naval Research Logistics (NRL)*, vol. 52, no. 7, pp. 668–681, 2005.

[48] M. Dahane, C. Clémentz, and N. Rezg, "Analysis of joint maintenance and production policies under a subcontracting constraint," *International Journal of Production Research*, vol. 46, no. 19, pp. 5393–5416, 2008.

[49] A. Berrichi, L. Amodeo, F. Yalaoui, E. Châtelet, and M. Mezghiche, "Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem," *Journal of Intelligent Manufacturing*, vol. 20, no. 4, pp. 389–400, 2009.

[50] A. Almuhtady, S. Lee, E. Romeijn, M. Wynblatt, and J. Ni, "Degradation-based swapping policy with application to fleet-level battery utilization," *Transportation Science-Special Issue on Energy and Transportation: Meeting the Challenge*, no. TS-2012-0013, Submitted in 2012 and accepted with changes in 2013.

[51] A. Almuhtady, A. Nasir, S. Lee, and J. Ni, "Stochastic degradation-based swapping policy with application to fleet-level battery utilization," *Expert Systems with Applications*, Submitted.

[52] I. Grossmann, S. V. D. Heever, and I. Harjunkoski, "Discrete optimization methods and their role in the integration of planning and scheduling," *AICHE Symposium Series*, vol. 98, no. 326, pp. 150–168, 2002.

[53] S. B. Petkov and C. D. Maranas, "Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty," *Industrial & Engineering Chemistry Research*, vol. 36, no. 11, pp. 4864–4881, 1997.

[54] D. B. Birewar and I. E. Grossmann, "Simultaneous production planning and

scheduling in multiproduct batch plants," *Industrial & Engineering Chemistry Research*, vol. 29, no. 4, pp. 570–580, 1990.

[55] V. Chvatal, *Linear programming*. Series of books in the mathematical sciences, New York: W. H. Freeman, 1983.

[56] K. Schittkowski, "The nonlinear programming method of Wilson, Han, and Powell with an augmented lagrangian type line search function," *Numerische Mathematik*, vol. 38, no. 1, pp. 115–127, 1982.

[57] O. K. Gupta and A. Ravindran, "Branch and bound experiments in convex nonlinear integer programming," *Management Science*, vol. 31, no. 12, pp. 1533–1546, 1985.

[58] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[59] A. M. Geoffrion, "Generalized benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, 1972.

[60] M. Duran and I. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical Programming*, vol. 39, no. 3, pp. 337–337, 1987.

[61] R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," *Mathematical Programming*, vol. 66, no. 1, pp. 327–349, 1994.

[62] I. Quesada and I. E. Grossmann, "An LP/NLP based branch and bound algorithm for convex MINLP optimization problems," *Computers & Chemical Engineering*, vol. 16, no. 10-11, pp. 937–947, 1992.

[63] T. Westerlund and F. Pettersson, "An extended cutting plane method for solving convex MINLP problems," *Computers & Chemical Engineering*, vol. 19, no. 0, pp. 131–136, 1995.

[64] G. R. Kocis and I. E. Grossmann, "Relaxation strategy for the structural optimization of process flow sheets," *Industrial & Engineering Chemistry Research*, vol. 26, no. 9, pp. 1869–1880, 1987.

[65] J. Viswanathan and I. E. Grossmann, "A combined penalty function and outer-approximation method for MINLP optimization," *Computers & Chemical Engineering*, vol. 14, no. 7, pp. 769–782, 1990.

[66] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. 1975.

[67] S.-F. Hwang and R.-S. He, "A hybrid real-parameter genetic algorithm for function optimization," *Advanced Engineering Informatics*, vol. 20, no. 1, pp. 7–21, 2006.

[68] J. Y. Suh and D. Van Gucht, *Distributed genetic algorithms*. Computer Science Department, Indiana Univ., 1987.

[69] C. G. Kirkpatrick, S. and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[70] D. Adler, "Genetic algorithms and simulated annealing: a marriage proposal," *IEEE International Conference on Neural Networks*, vol. 2, no. 1, pp. 1104–1109, 1993.

[71] Z. Yang, D. Djurdjanovic, and J. Ni, "Maintenance scheduling in manufacturing systems based on predicted machine degradation," *Journal of Intelligent Manufacturing*, vol. 19, no. 1, pp. 87–98, 2008.

[72] P. Y. Papalambros and D. J. Wilde, *Principles of optimal design: modeling and computation*. Cambridge university press, 2000.

[73] P. M. Verderame, J. A. Elia, J. Li, and C. A. Floudas, "Planning and scheduling under uncertainty: A review across multiple sectors," *Industrial & Engineering Chemistry Research*, vol. 49, no. 9, pp. 3993–4017, 2010.

[74] Z. Li and M. Ierapetritou, "Process scheduling under uncertainty: Review and challenges," *Computers & Chemical Engineering*, vol. 32, no. 4, pp. 715 – 727, 2008.

[75] T. Gal and J. Nedoma, "Multiparametric linear programming," *Management Science*, vol. 18, no. 7, pp. 406–422, 1972.

[76] J. Acevedo and E. N. Pistikopoulos, "An algorithm for multiparametric mixed-integer linear programming problems," *Operations Research Letters*, vol. 24, no. 3, pp. 139 – 148, 1999.

[77] V. Dua, N. A. Bozinis, and E. N. Pistikopoulos, "A multiparametric programming approach for mixed-integer quadratic engineering problems," *Computers & Chemical Engineering*, vol. 26, no. 4, pp. 715 – 733, 2002.

[78] S. Orun, . K. Altinel, and ner Hortasu, "Scheduling of batch processes with operational uncertainties," *Computers & Chemical Engineering*, vol. 20, Supplement 2, no. 0, pp. S1191 – S1196, 1996.

[79] X. Lin, S. L. Janak, and C. A. Floudas, "A new robust optimization approach for scheduling under uncertainty:: I. bounded uncertainty," *Computers & Chemical Engineering*, vol. 28, no. 6, pp. 1069 – 1085, 2004.

[80] S. L. Janak, X. Lin, and C. A. Floudas, "A new robust optimization approach for scheduling under uncertainty: II. uncertainty with known probability dis-

tribution," *Computers & Chemical Engineering*, vol. 31, no. 3, pp. 171 – 195, 2007.

[81] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.

[82] P. H. Zipkin, "Foundations of inventory management," 2000.

[83] P. A. Jensen and J. F. Bard, *Operations research: models and methods*, vol. 1. John Wiley & Sons Inc, 2003.

[84] R. Spotnitz, "Simulation of capacity fade in lithium-ion batteries," *Journal of Power Sources*, vol. 113, no. 1, pp. 72–80, 2003.

[85] L. Srichandan Mishra, U. Misra, and G. Mishra, "A deterministic inventory model for deteriorating items with on-hand inventory dependent, variable type demand rate," *International Journal of Research and Reviews in Applied Sciences*, vol. 7, no. 2, 2001.

[86] S. Goyal and B. Giri, "Recent trends in modeling of deteriorating inventory," *European Journal of Operational Research*, vol. 134, no. 1, pp. 1 – 16, 2001.

[87] T. M. Whitin, *The theory of inventory management*. Greenwood Press, 1970.

[88] K. Cobbaert and D. V. Oudheusden, "Inventory models for fast moving spare parts subject to sudden death obsolescence," *International Journal of Production Economics*, vol. 44, no. 3, pp. 239 – 248, 1996.

[89] P. Ghare and G. Schrader, "A model for exponentially decaying inventory," *Journal of Industrial Engineering*, vol. 14, no. 5, pp. 238–243, 1963.

[90] S. Nahmias, "Perishable inventory theory: A review," *Operations Research*, vol. 30, no. 4, pp. 680–708, July/August 1982.

[91] F. Raafat, "Survey of literature on continuously deteriorating inventory models," *The Journal of the Operational Research Society*, vol. 42, no. 1, pp. pp. 27–37, 1991.

[92] B. Karmakar and K. D. Choudhury, "A review on inventory models for deteriorating items with shortages," *Assam University Journal of Science and Technology*, vol. 6, no. 2, pp. 51–59, 2010.

[93] P. Nandakumar and T. E. Morton, "Near myopic heuristics for the fixed-life perishability problem," *Management Science*, vol. 39, no. 12, pp. 1490–1498, 1993.

[94] L. Liu and Z. Lian, "(s, S) continuous review models for products with fixed lifetimes," *Operations Research*, vol. 47, no. 1, pp. 150–158, 1999.

[95] S. Kalpakam and K. Sapna, "Continuous review (s, S) inventory system with random lifetimes and positive leadtimes," *Operations Research Letters*, vol. 16, no. 2, pp. 115–119, 1994.

[96] L. Liu, "(s, S) continuous review models for inventory with random lifetimes," *Operations Research Letters*, vol. 9, no. 3, pp. 161–167, 1990.

[97] L. Liu and D.-H. Shi, "An (s, S) model for inventory with exponential lifetimes and renewal demands," *Naval Research Logistics (NRL)*, vol. 46, no. 1, pp. 39–56, 1999.

[98] L. Liu and T. Yang, "An (s,S) random lifetime inventory model with a positive lead time," *European Journal of Operational Research*, vol. 113, no. 1, pp. 52 – 63, 1999.

[99] N. Ravichandran, "Stochastic analysis of a continuous review perishable inven-

tory system with positive lead time and poisson demand," *European Journal of operational research*, vol. 84, no. 2, pp. 444–457, 1995.

[100] G. J. J. Van Zyl, *Inventory control for perishable commodities*. PhD thesis, University of North Carolina at Chapel Hill, 1963.

[101] K. J. Heng, J. Labban, and R. J. Linn, "An order-level lot-size inventory model for deteriorating items with finite replenishment rate," *Computers & Industrial Engineering*, vol. 20, no. 2, pp. 187 – 197, 1991.

[102] F. F. Raafat, P. M. Wolfe, and H. K. Eldin, "An inventory model for deteriorating items," *Computers & Industrial Engineering*, vol. 20, no. 1, pp. 89–94, 1991.

[103] C.-H. Goh, B. S. Greenberg, and H. Matsuo, "Two-stage perishable inventory models," *Management Science*, vol. 39, no. 5, pp. 633–649, 1993.

[104] H. Xu and H.-P. Wang, "An economic ordering policy model for deteriorating items with time proportional demand," *European Journal of Operational Research*, vol. 46, no. 1, pp. 21–27, 1990.

[105] L. Benkherouf, "On an inventory model with deteriorating items and decreasing time-varying demand and shortages," *European Journal of Operational Research*, vol. 86, no. 2, pp. 293–299, 1995.

[106] Z. T. Balkhi and L. Benkherouf, "On the optimal replenishment schedule for an inventory system with deteriorating items and time-varying demand and production rates," *Computers & industrial engineering*, vol. 30, no. 4, pp. 823–829, 1996.

[107] B. Giri, A. Goswami, and K. Chaudhuri, "An EOQ model for deteriorating

items with time varying demand and costs," *Journal of the Operational Research Society*, pp. 1398–1405, 1996.

[108] M. Hariga, "Optimal EOQ models for deteriorating items with time-varying demand," *Journal of the Operational Research Society*, pp. 1228–1246, 1996.

[109] A. Pal and B. Mandal, "An EOQ model for deteriorating inventory with alternating demand rates," *Journal of Applied Mathematics and Computing*, vol. 4, no. 2, pp. 397–407, 1997.

[110] A. Andijani and M. Al-Dajani, "Analysis of deteriorating inventory/production systems using a linear quadratic regulator," *European journal of operational research*, vol. 106, no. 1, pp. 82–89, 1998.

[111] T. Chakrabarty, B. Giri, and K. Chaudhuri, "An EOQ model for items with weibull distribution deterioration, shortages and trended demand: an extension of philip's model," *Computers & Operations Research*, vol. 25, no. 7, pp. 649–657, 1998.

[112] Z. T. Balkhi, "On the global optimal solution to an integrated inventory system with general time varying demand, production and deterioration rates," *European Journal of Operational Research*, vol. 114, no. 1, pp. 29–37, 1999.

[113] P. S. Deng, R. H.-J. Lin, and P. Chu, "A note on the inventory models for deteriorating items with ramp type demand rate," *European Journal of Operational Research*, vol. 178, no. 1, pp. 112–120, 2007.

[114] K. Skouri, I. Konstantaras, S. Papachristos, and I. Ganas, "Inventory models with ramp type demand rate, partial backlogging and weibull deterioration rate," *European Journal of Operational Research*, vol. 192, no. 1, pp. 79–92, 2009.

[115] L. Liu and K. Cheung, "Service constrained inventory models with random lifetimes and lead times," *Journal of the Operational Research Society*, vol. 48, no. 10, pp. 1022–1028, 1997.

[116] L. Aggoun, L. Benkherouf, and L. Tadj, "A hidden markov model for an inventory system with perishable items," *International Journal of Stochastic Analysis*, vol. 10, no. 4, pp. 423–430, 1997.

[117] L. Aggoun, L. Benkherouf, and L. Tadj, "Optimal adaptive estimators for partially observed numbers of defective items in inventory models," *Mathematical and computer modelling*, vol. 29, no. 6, pp. 83–93, 1999.

[118] D. E. Shobrys and D. C. White, "Planning, scheduling and control systems: why cannot they work together," *Computers & chemical engineering*, vol. 26, no. 2, pp. 149–160, 2002.

[119] M. Dror, M. Ball, and B. Golden, "A computational comparison of algorithms for the inventory routing problem," *Annals of Operations Research*, vol. 4, no. 1, pp. 1–23, 1985.

[120] M. Dror and M. Ball, "Inventory/routing: Reduction from an annual to a short-period problem," *Naval Research Logistics (NRL)*, vol. 34, no. 6, pp. 891–905, 1987.

[121] A. J. Kleywegt, V. S. Nori, and M. W. Savelsbergh, "The stochastic inventory routing problem with direct deliveries," *Transportation Science*, vol. 36, no. 1, pp. 94–118, 2002.

[122] A. J. Kleywegt, V. S. Nori, and M. W. Savelsbergh, "Dynamic programming approximations for a stochastic inventory routing problem," *Transportation Science*, vol. 38, no. 1, pp. 42–70, 2004.

[123] L. M. Hvattum and A. Løkketangen, "Using scenario trees and progressive hedging for stochastic inventory routing problems," *Journal of Heuristics*, vol. 15, no. 6, pp. 527–557, 2009.

[124] L. Bertazzi, A. Bosco, F. Guerriero, and D. Laganà, "A stochastic inventory routing problem with stock-out," *Transportation Research Part C: Emerging Technologies*, 2011.

[125] G. Parlier, *Transforming US Army supply chains: strategies for management innovation*. Business Expert Press, 2011.

[126] G. H. Parlier, "Transforming a complex, global organization: Operations research and management innovation for the us armys materiel enterprise," in *International Conference on Operations Research and Enterprise Systems*, (Barcelona, Spain), Feb 2013.

[127] Nationwide-Lifts, "Commercial elevators," 2012.

[128] A. S. of Heating, Refrigerating, and A.-C. Engineers, *ASHRAE Handbook. Heating, Ventilating, and Air-conditioning Applications*. American Society of Heating Refrigerating and Air-Conditioning Engineers, 1991.

[129] A. Hemmerly-Brown, "Arforgen: Army's deployment cycle aims for predictability," *Army Millitary*, 2009.