NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF MARKOV CHAIN MONTE CARLO METHODS IN MULTI-INDENTURE INVENTORY OPTIMIZATION

by

Adam M. Alleman

September 2022

Co-Advisors: Ruriko Yoshida
Jefferson Huang
Second Reader: Javier Salmeron-Medrano

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
ANALYSIS OF MARKOV CHAIN MONTE CARLO METHODS IN MULTI-INDENTURE INVENTORY OPTIMIZATION

5. FUNDING NUMBERS

6. AUTHOR(S) Adam M. Alleman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NAVSUP WSS, Philadelphia, PA 19130

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

U.S. Navy aircraft are required to meet minimum operational availability targets, while minimizing spare parts procurement costs. The current optimization model written by Salmeron and Buss, uses marginal analysis, as described by Sherbrooke, to determine optimal sparing policies for this highly complex multi-indenture model. The literature lacks alternative optimization methodologies for such a problem, so we propose an alternative approach utilizing simulated annealing (SA), a Markov Chain Monte Carlo algorithm. We present three SA approaches tested in three case studies of varying size and complexity. Our initial findings show that in very simple problems, SA is easily capable of outperforming marginal analysis; however, problems with more complexity have large optimality gaps. This is likely because the SA Markov chain is unable to effectively explore the multi-indenture structure of the problem. We implement a method to account for this structure that intelligently builds initial feasible solutions using an epsilon-greedy approach to marginal analysis. This approach produces better results than NAVARM in more than half of the trials on problems of moderate complexity. We also implement a novel method for calculating operational availability that may allow full scale problems to be optimized more efficiently.

14. SUBJECT TERMS Markov Chain Monte Carlo, MCMC, Metropolis-Hastings, simulated annealing, stock level optimization, multi-indenture optimization, readiness based sparing, sparing policy, NAVARM, aviation	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ANALYSIS OF MARKOV CHAIN MONTE CARLO METHODS IN MULTI-INDENTURE INVENTORY OPTIMIZATION

Adam M. Alleman
Lieutenant Commander, United States Navy
BSCE, Citadel, Military College of South Carolina, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: Ruriko Yoshida
Co-Advisor

Jefferson Huang
Co-Advisor

Javier Salmeron-Medrano
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

U.S. Navy aircraft are required to meet minimum operational availability targets, while minimizing spare parts procurement costs. The current optimization model written by Salmeron and Buss, uses marginal analysis, as described by Sherbrooke, to determine optimal sparing policies for this highly complex multi-indenture model. The literature lacks alternative optimization methodologies for such a problem, so we propose an alternative approach utilizing simulated annealing (SA), a Markov Chain Monte Carlo algorithm. We present three SA approaches tested in three case studies of varying size and complexity. Our initial findings show that in very simple problems, SA is easily capable of outperforming marginal analysis; however, problems with more complexity have large optimality gaps. This is likely because the SA Markov chain is unable to effectively explore the multi-indenture structure of the problem. We implement a method to account for this structure that intelligently builds initial feasible solutions using an epsilon-greedy approach to marginal analysis. This approach produces better results than NAVARM in more than half of the trials on problems of moderate complexity. We also implement a novel method for calculating operational availability that may allow full scale problems to be optimized more efficiently.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AVCAL	aviation consolidated allowance list
CBO	Congressional Budget Office cumulative
CDF	density function
CNA	Center for Naval Analysis
CNO	Chief of Naval Operations
CTMC	continuous time Markov chain
CVN	aircraft carrier
DoD	Department of Defense
DoN	Department of the Navy
EBO	expected backorder
FMC	fully mission capable
IMF	intermediate maintenance facility
LHA	amphibious assault ship
MALS	Marine Aviation Logistics Squadron
MALSP	Marine aviation logistics support package
MC	mission capable
MCAS	Marine Corps Air Station
MCMC	Markov Chain Monte Carlo

METRIC Multi-Echelon Technique for Recoverable Item

MILSPEC military specification

NAS Naval Air Station

NAVARM Naval Aviation RBS Model

NAVSUP Naval Supply Systems Command

NMC non-mission capable

NSAM NAVARM Simulated Annealing Model

NSN national stock number

NSS Naval Sustainment System

OPNAV Office of the Chief of Naval Operations (CNO)

P2P Performance to Plan

PMC partially mission capable

RBS readiness based sparing

SASS supplemental aviation supply support

SHORCAL shore-based consolidated allowance list

SRA shop replaceable assembly

TMS type/model/series

USN United States Navy

VBA Visual Basic for Applications

VTMR variance-to-mean ratio

WRA weapons replaceable assembly

WS weapon system

WSS Naval Supply Systems Command (NAVSUP) Weapon System Support

Executive Summary

Keeping the fleet of U.S. Naval aircraft ready for combat rests in the balance of precise preventative and corrective maintenance requirements, personnel management and the optimal distribution of required spare parts. In 2018, responding to undesirably low readiness levels amongst fleet aircraft, Secretary of Defense James Mattis ordered that the Navy's fleet of F/A-18 Hornets and Super Hornets obtain and maintain an 80% level of operational availability, A_o. This metric of readiness is a composite measure of the total proportion of time these aircraft are capable of performing at least one mission set it has been assigned. In 2022, Commander Naval Air Forces adopted this readiness goal for all fleet aircraft (Katz 2022). There has been a plethora of effort expended in the optimization of maintenance timelines and personnel management that have led to significant readiness improvements. Among these has been a concerted effort by NAVSUP Weapon System Support (WSS) to improve the optimization of the RBS model that is used to outfit spare parts allowances to Naval Aviation sites across the enterprise.

Naval Aviation inventory optimization uses a multi-indenture model where each weapon system (WS)'s A_o is modeled as the product of the A_o of each of its top level part types, called a weapons replaceable assembly (WRA). The A_o for each WRA is a function of the number of expected backorders (EBOs) of the parts sub-indentured to it as a result of a given stocking policy. This structure creates a classically intractable optimization problem where the optimal cost stocking policy must be generated by a recursive and iterative methodology of selecting allowances for each part type. The Navy's current optimization model, NAVARM, authored by Salmeron and Buss (2021), utilizes a methodology proposed by Sherbrooke (2004) called marginal analysis to build optimal sparing policies. This creates a "shopping list" of greedy decisions where the allowances of part types are increased one-by-one until a feasible solution is found.

The results of our research serve as a validation of the suitability of the marginal analysis approach for solving the RBS optimization problem. However, given the immense size and complexity of the decision space created by a problem with potentially tens of thousands of decision variables that are all inter-dependent, the probability of any particular optimization method finding the true global optimum such a problem is very small. We take this to

mean that, while local optima determined by marginal analysis are possibly near-optimal, there could be many feasible solutions that can be found that improve the solution while remaining feasible.

Our approach to the RBS problem is a method based on the work of Metropolis et al. (1953) that came to be known as Markov Chain Monte Carlo (MCMC). This method, further refined by Kirkpatrick et al. (1983) to a system now called simulated annealing, randomly selects stocking policies based on previously accepted feasible policies in an iterative fashion, where the probability of accepting any new decision is a function of the costs of the currently accepted solution and randomly generated neighbor solution, as well as the feasibility of the neighbor solution. This simulated annealing approach allows for a "cooling" process that provides for wide exploration of the decision space in the early portions of each run and for deeper exploitation of better regions of the feasible decision space in the later portions of the run. Our simulated annealing algorithm, NSAM, of which we propose three versions, is evaluated on three different multi-indenture problems of increasing complexity. As a result of the intractability of determining the global optimal solution for all but the most simple of such problems, we compare our results to the incumbent algorithm's, NAVARM, solutions.

Our results first demonstrate that an MCMC approach is able to outperform NAVARM's marginal analysis approach on the simplest problem set; however, as the problem's complexity increases, a significant optimality gap becomes apparent that a simple simulated annealing algorithm is unable to overcome. Our research shows that this gap is a result of the random selection of part types that a simple MCMC method performs when changing allowances for the next neighbor policy to accept or reject. Our work indicates multi-indenture structure tends to favor the addition of expensive high level parts, i.e. WRAs, as these parts reduce EBO the most (thereby increasing A_o); therefore, it becomes very unlikely that a basic MCMC method will select the correct subset of a WRA's sub-indentured parts to increase allowance for if the parent WRA's allowance is decreased.

This complexity leads us to the creation of two more versions of the NSAM model that leverage what we know of the multi-indenture structure of the RBS problem. Our final approach leverages an epsilon-greedy approach to marginal analysis that creates initial feasible solutions by using the same methodology already resident in the NAVARM algorithm with a minute level of stochasticity introduced. We show that this approach creates solutions

that are both better than simple marginal analysis and solutions that can be improved by simulated annealing to also be closer to global optimal than those produced by NAVARM's base marginal analysis approach alone.

While we are able to prove improvement upon NAVARM sub-optimal solutions using the most basic toy RBS problem and a very basic real-world RBS problem, we find that computation times for full scale RBS problems such as those required for the outfitting of CVN air wings, are beyond reasonable with results that are mixed under the best circumstances. To address this, we also propose a new methodology to calculating A_o that, by using matrix operations versus iterative loops, leverages the architecture of today's graphics processing units to determine the availability of any given weapon system as a vector of the limiting probabilities of a continuous time Markov chain (CTMC). We have not validated this approach to A_o calculation, however we provide methodology for its implementation as a recommendation for future research to accompany and increase the efficiency of the use of MCMC optimization methods.

Based on this work, we confidently recommend simulated annealing as a method to refine locally optimal solutions to small scale RBS problems determined via other means, such as marginal analysis or an epsilon-greedy approach. Furthermore, we are confident that once calculation of A_o is further optimized this methodology will be readily adapted to the refinement of any multi-indenture inventory optimization model.

List of References

Katz J (2022) Navy air boss has new aircraft readiness targets to hit. Breaking Defense, URL https://breakingdefense.com/2022/02/navy-air-boss-has-new-aircraft-readiness-rate-north-stars-to-follow/.

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.

Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6):671–680.

Salmeron J, Buss A (2021) Project deliverable: Naval aviation readiness-based sparing model – release 2.6. Naval Postgraduate School, Monterey, CA, provided by J. Salmeron with approval from NAVSUP-WSS Philadelphia.

Sherbrooke C (2004) *Optimal Inventory Modeling of Systems, Multi-Echelon Techniques, Second Edition* (Kluwer Academic Publishers, Boston).

Acknowledgments

I would like to first thank my grandfather and my grandmother for giving me the tools I needed to make it this far in life and find some modicum of success along the way, I miss you every day. I would also like to thank my mother and my father for their support and encouragement though-out my career and my time here at NPS.

I would also like to thank my other half, Crystal for her non-stop cheerleading, encouragement and advice. You have helped keep me sane and put up with way more of my frustrated moments than anyone should ever have to. Thank you so much for being there, even when that meant us sitting quietly for endless hours working on our own operations research homework while putting up with our beautifully annoying kitties (Buttons and Little, I thank you too). I love you and I thank you from the bottom of my heart.

I would be remiss if I did not mention the awesome folks in my cohort, our jokes are our own and they should probably never see the light of day. Thanks guys for all the friendship, great times and the endless commiseration.

To the folks at NAVSUP WSS, thank you so much for your time and energies in making this work possible. In particular I would like to thank Dan Martinez and Karla Eastburg for staying interested in my project and being my open line of communication at WSS, you guys are great!

Last and certainly not least, I thank the fantastic faculty and staff of the NPS Operations Research Department. Specifically my advisors, Dr. Yoshida and Dr. Huang, you guys were the best sounding board a grad student could ask for! Thank you so much for your guidance, patience and especially for reigning me in when I wanted to bite off more than I could chew. To my second reader, Dr. Salmeron, your classroom instruction was my initial inspiration for this work. I cannot thank you enough for keeping your door open for me and readily answering my frequent questions or simply chatting about the topic du jour.

Thank you everyone, I certainly could never have done any of this without your help!

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

In fiscal year 2022, the Naval Aviation enterprise is projected to spend $2.82B on procurement or repair of aviation related equipment (Department of the Navy 2021). This money effectively drives the readiness of fleet aircraft, so it is crucial that every last ounce of effectiveness is squeezed from the dollars that are spent. In 2018 Secretary of Defense James Mattis issued a memorandum that required the Navy's fleet of F/A-18 Hornets and Super Hornets must meet a minimum A_o target of 80% mission capable by the end of Fiscal Year 2019 (Mehta 2018). Following this example in 2022, Commander Naval Air Forces has adopted the same readiness goal for all fleet aircraft (Katz 2022). In order to meet this goal, the Navy has been working on an all angles approach that incorporates all elements of the logistics apparatus ranging from maintenance procedures and manning requirements to increases in funding used for sustainment.

Much effort has gone into modernizing maintenance practices under Office of the Chief of Naval Operations (CNO) (OPNAV) directed programs such as Performance to Plan (P2P) and Naval Sustainment System (NSS). A press release from OPNAV (2020) indicated that the F/A-18 availability goal had been reached using data driven solutions that include the use of machine-learning models "for determining the monthly number of mission capable jets per squadron by incorporating manning-training-equipment datasets." With such success evinced by employing advanced data analytics on the maintenance side of the enterprise, it becomes a natural progression to attempt to do the same for the supply chain side.

Today, readiness is measured in terms of operational availability (A_o). In rough terms A_o is the fraction of time an aircraft (and by extension a fleet of aircraft) is capable of executing an assigned mission. This is determined by dividing the amount of time an aircraft is mission capable by the entire time horizon considered. There are some differences in how this is calculated at the congressional level and within the Department of Defense (DoD) (Congressional Budget Office 2022), which has led to some disagreement on what constitutes readiness. The Congressional Budget Office (CBO) measures readiness across all aircraft in the Navy's inventory, Department of the Navy (DoN) reports readiness based

on aircraft actually assigned to operational squadrons (Katz 2022). The effect of this is that any aircraft in storage or in depot maintenance counts against overall readiness as measured by the CBO through a reduction in the numerator of the above-described A_o calculation. By comparison, the numerator would remain unchanged in the DoN calculation.

Regardless of how it is calculated, the numerator of the A_o equation is dependent on the amount of time spent in maintenance while the denominator is simply the cumulative time considered regardless of aircraft status. An aircraft can be placed in a maintenance status for periodic preventive maintenance and for corrective maintenance. As mentioned above, P2P and NSS have already addressed and made significant improvements in maintenance processes themselves. That means that there is potentially room for improving elements of the logistics process that impact the number of times an aircraft ends up in the maintenance cycle. It is in this vein that we initially attempted a machine learning approach to the spare parts allowance optimization problem.

The Naval Aviation spare parts inventory optimization problem is one of unique complexity. It can be modeled as a multi-indenture problem that spans different aircraft, hereafter referred to as weapon system (WS). The term "multi-indenture" implies parent-child relationships, where a given part is actually an assembly consisting of multiple sub-assemblies in a tree-like structure of sub-indenture where the most complex assembly, called a weapons replaceable assembly (WRA), is at the top of a hierarchy of sub-assemblies, shop replaceable assembly (SRA). These WRAs can themselves be comprised of sub-indentured SRAs. To make things more complicated, a WRA or SRA on one weapon system has no guarantee of having the same sub-indenture structure as any other instance of that part type on the aircraft.

Our initial efforts to utilize reinforcement learning, in particular Q-learning, to approach the problem quickly led us to conclude that it is inappropriate for our multi-indenture inventory optimization problem. The very basis of an inventory optimization problem presumes that a stocking policy is set first and its effectiveness, as measured by A_o, is analyzed after the fact either by real world observation, calculation of theoretical expectation or by advanced simulations over long time horizons. Agent based approaches make little sense in this context as the solution is the result of a single decision and not a sequence thereof. The problem is that of selecting one solution within a decision space comprised of potentially

tens of thousands of decision variables. Seeing the problem though this lens brought us to the idea of using a Markov Chain Monte Carlo (MCMC) approach as first proposed by Metropolis (Metropolis et al. 1953).

1.1 Problem Statement

The problem of multi-indenture inventory optimization is approached in detail in Sherbrooke's Optimal Inventory Modeling of Systems 2004. One of the important solution heuristics for this problem is a process he calls marginal analysis. It is on this text that Salmeron and Buss based the "greedy heuristic" that works at the heart of their optimization model (Salmeron and Buss 2021) called Naval Aviation readiness based sparing (RBS) Model (NAVARM). Naval Supply Systems Command (NAVSUP) Weapon System Support (WSS) uses NAVARM to optimize aviation spare parts allowances at single site locations. These sites range from afloat inventory points such as aircraft carrier (CVN) and amphibious assault ships (LHAs) to large ashore sites at Naval Air Stations (NASs) and Marine Corps Air Stations (MCASs).

The current NAVARM model is accredited by OPNAV and provides exceptional allowance optimization. However, given the complexity of the decision space and the methodology underpinning marginal analysis, there is no guarantee that NAVARM solutions are globally optimal. In fact, the opposite is almost assuredly true though NAVARM solutions have been proven to be adequate to justify the obligation of tens of millions in taxpayer dollars that for which they call.

The complexity of the Naval Aviation allowancing problem extends beyond the multitude of decision variables representing the quantity of spares for each part type in a WS's aviation consolidated allowance list (AVCAL). As a multi-indenture problem that shares common parts across platforms at differing levels of indenture, with differing sub-indenture structures, the problem structure implies an intractably large quantity of potential solutions. NAVARM's marginal analysis approach is exceptionally well suited for this problem type, but its results have not yet been challenged by methodologies made possible by large scale parallel computing offered by today's supercomputing clusters. Our research seeks to bridge this gap by proposing the use of MCMC as an optimization method for the multi-indenture inventory problem.

1.2 Purpose of the Study

The focus of our study is on developing an alternative approach to solving the highly complex multi-indenture inventory optimization problem. By formulating an MCMC algorithm and coding in a computational language for ingesting site data and producing alternative stocking policies, we hope to provide a basis for analyzing the effectiveness of both the incumbent model, NAVARM, and our new MCMC-based model we have named the NAVARM Simulated Annealing Model (NSAM). Improvement upon or confirmation of the optimality offered by NAVARM's marginal analysis approach is the underlying motivation for which this research.

1.3 Research Questions

1.3.1 Conceptual Questions

Is an MCMC algorithm capable of providing solutions that approximate a given site's globally cost-optimal stocking policy? If so, are these solutions attainable in a reasonable amount of time? If MCMC methods are not effective, is there an underlying structure that can be identified as being prohibitive to success? In most cases, on a long enough timeline, an algorithm which randomly explores the entire solution space will find a globally optimum solution since the state space is finite. However, the computational time required for systems as complex as we approach is prohibitively expensive. Therefore, a locally optimal solution must be settled upon that is reached using stopping conditions that allow the algorithm to terminate within a time frame that is reasonable in light of the user's needs.

1.3.2 Thesis Questions

In a multi-indenture inventory optimization problem, can an MCMC algorithm perform as well or better than the marginal analysis approach used in NAVARM? Measures of effectiveness necessarily include the following:

1. Quality as measured by objective function value, or total cost of the output policy,
2. Quality as measured by slack remaining in problem A_o constraints,
3. Performance as measured by computational resources required to arrive at a solution.

1.4 Research Approach

This study was undertaken in a multi-stage approach that allowed for iterative creation and refinement of the NSAM algorithm. In its three final versions, NSAM, which was developed in Java (Oracle 2014), utilized key components from NAVARM to both ingest configuration data and calculate A_o that were configured for multithreaded operation to be able to run many instances in parallel. This design allows us to set up experiments to tweak NSAM's parameters to determine what parameters were effective in achieving better solutions vs. those which offered little more than a trap for valuable computing resources.

NSAM began life as a simple simulated annealing algorithm coded in Java where a handful of parts were created that contributed a deterministic "amount of A_o" which closely resembled a typical knapsack type problem. A_o constraint conditions were measured as a simple sum of each part's assigned A_o contribution multiplied by the quantity of that part selected for the stocking policy. Results of this algorithm were compared to results where the same parts were fed into an optimization program coded using Python Pyomo, Bynum et al. (2021) and Hart et al. (2011), using the Gurobi solver (Gurobi Optimization, LLC 2022). Once output from NSAM matched the Gurobi solution, development moved to the next stage.

The next stage transitioned the calculation of A_o to that of a limiting distribution for a continuous time Markov Chain. The structure of this problem was limited to two WRAs each with two SRA part types. This limitation was made to ease the computation of each WRA's vector of limiting probabilities; though the addition of more complex structure is possible, we do not discuss it at length. It should be noted that this treatment of the A_o calculation is a departure from the literature and merits further investigation as an alternative to Sherbrooke's equations (Sherbrooke 2004). Such a method offers the potential for computational resource optimization by approaching the problem using high dimensional matrices for which graphics processing units and tensor processing units are architecturally optimized.

With this framework, through which we could treat A_o as a stochastic variable, we were able to move on to the final stages of NSAM's development followed by experimentation and analysis presented in this thesis.

1.5 Significance to the Field

Currently, there does not appear to be any attempts to optimize a multi-indenture inventory problem using MCMC methods such as the ones presented in this thesis. The goal of this research is to provide the following:

1. Determine whether an MCMC-based methodology can improve on the current practice of using marginal analysis.
2. Determine impediments that the underlying structure of such a problem could pose for future work in the field.
3. Provide a novel methodology for computing A_o that can reduce resource requirements in future full scale implementations.
4. Develop a baseline algorithm for use in the field that can be improved upon in the future that does not require extensive tailoring in order to employ different measures of readiness.

1.6 Thesis Structure

We begin in Chapter 2 by presenting the current state of the art. We first discuss the Naval Aviation Maintenance cycle to provide adequate context for the RBS problem. Following this are brief overviews of the literature supporting the current optimization model, NAVARM, including a discussion of the VARI-METRIC model. We conclude Chapter 2 with details on the functionality of NAVARM and simulated annealing generally.

Chapter 3 introduces the NSAM algorithm, beginning with how the data is sourced, processed and routed through the algorithm. We then present three versions of NSAM and how they were scaled from the most basic possible form to the final versions. Each version of NSAM behaves differently as distinct approaches to the RBS problem, so we provide detailed looks into each of the three algorithms.

Chapter 4 discusses the results of experimentation with the three NSAM versions. This discussion is broken into three parts, where NSAM's efficacy is analyzed against three problems of increasing complexity. As the development of NSAM is very iterative, so too are the lessons learned at each of the three levels of complexity.

We conclude with a summary of our findings. Also discussed are limitations on the scope of

our work and the applicability of our findings. Finally we offer recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background and Literature Review

The Naval Aviation enterprise models resupply of spare parts using a single-site[1] , multi-indenture, VARI-METRIC model (Slay 1980). In this model, failures of individual parts are assumed to occur according to a Poisson Process but the aggregate expected failures of large assemblies of parts are modeled using the negative binomial distribution. This model is used in the wider context of RBS to target desired A_o goals using the NAVARM algorithm to generate near-optimal stocking policies[2] using marginal analysis (Gross, as cited by Sherbrooke (2004 30).

2.1 The Naval Aviation Maintenance Cycle

A basic understanding of repairable parts recovery and resupply within the Naval Aviation enterprise is critical to developing intuition needed to understand the inventory optimization methodologies we discuss in this thesis.

The cycle starts when a part fails and requires repair. The inoperable part is removed from the aircraft[3] by squadron maintainers and turned in to the intermediate maintenance facility (IMF) where it is inducted into the repair cycle. At this a point, a determination is made whether the part can be repaired at the IMF or if it must be sent off for depot repair. If a spare part is in stock at the local supply department, it is immediately given to the squadron maintainers to return the aircraft to operation. If a spare is not available, a backorder is created. Backorders can be filled by completing the repair of the originally broken part or another identical part that was already inducted into the repair cycle. Conversely, if it is determined that the part is beyond capable maintenance at the IMF and the part is not carried in stock or is carried but there are not currently any spares on hand, then a requisition is created to fill the backorder. In any case, if a replacement part is issued from stock, the

[1] A "site" refers to a single aircraft carrier CVN, NAS or Marine Aviation Logistics Squadron (MALS) where it can be assumed that all aircraft stationed at that site service all maintenance requirements.

[2] Stocking policies are vectors of integer values corresponding to the quantity of spare parts for a paired vector of part types.

[3] This process also applies to ground support equipment and test equipment, though the allowancing process is generally treated differently.

stock level deficiency must be corrected either by receiving a replacement requisition from the Naval stock system or by repair of the original part and return to supply. A simplified graphical representation of this process is provided in Figure 2.1:

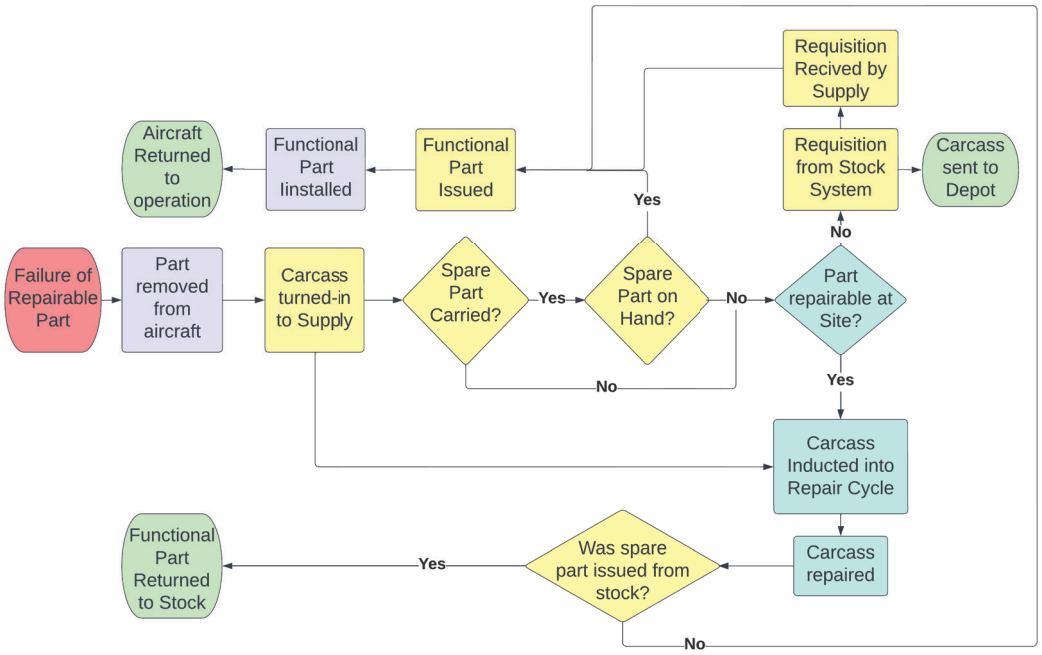

Figure 2.1. The Naval Aviation Repair Cycle: Green and red represent initial and final termini respectively, Purple indicates squadron actions, Yellow indicates Supply Department actions, Blue indicates IMF actions.

2.2 Multiple Indenture

Sherbrooke (2004) describes the principal concept of a multiple indenture model as an "engineering parts hierarchy" where there is a parent-child structure between the parts that comprise a WS. For ease of understanding and to reflect the lexicon used by Salmeron and Buss in NAVARM (2021), we must make a distinction between a part type and an instance of a part. The Navy uses national stock numbers (NSNs) to identify a part that meets a particular military specification (MILSPEC) . We will refer to any part that has the same NSN as a "part type". On a particular WS, there can be many different instances where

the same part type is used in different locations; we refer to these individual instances of a particular part type as a "candidate."

In Navy parlance, the candidates at the highest level of indenture are called WRAs. Sub-indentured, or child candidates to these WRAs, called SRAs have sub-indentured candidates, SSRAs, that in turn have sub-indentured candidates, SSSRAs, and so on to a typical maximum of 5 levels of sub-indenture. A simplistic example of this structure is shown in Figure 2.2.

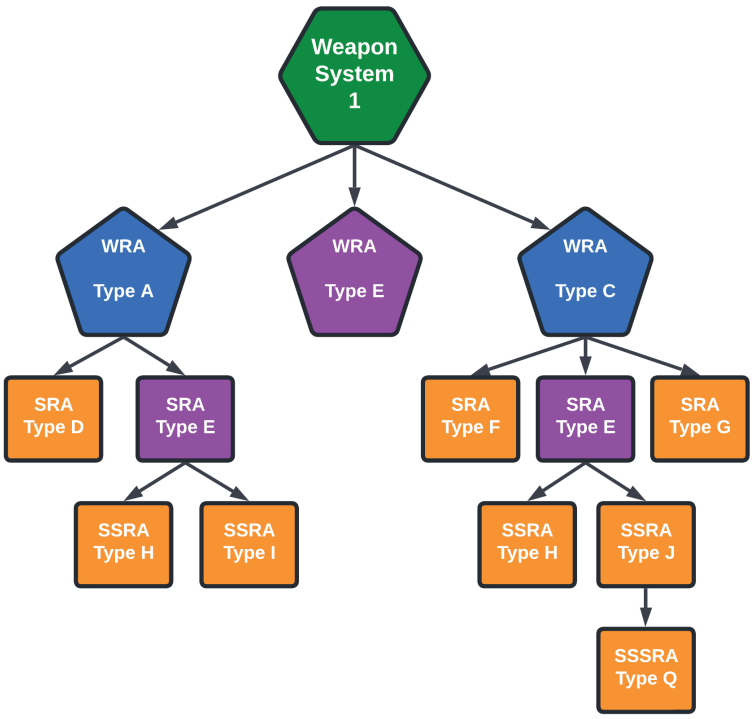

Figure 2.2. Example multiple-indenture structure with three WRA (A, E, C).

As seen in Figure 2.2, not all candidates have children or parents and not all candidates of the same part type (part type E in Figure 2.2 for example) have the same sub-indenture structure. The differences in sub-indenture structure between candidates of the same part type arise from physical characteristics of the WS involved. For instance, a certain type of hydraulic pump assembly may be completely accessible on the port side of an aircraft, but

due to the installation location of a communications antenna of on the starboard side, only certain sub-assemblies of the pump on that side may be removed without removal of the entire pump.

Generally speaking, candidates that are sub-indentured are less complex and, consequently, less costly than their parent candidate. This means that it is usually most advantageous to repair a downed aircraft by replacing the candidate at the lowest possible level of indenture. This is not universally true as there are instances where knowledge of the system by maintenance personnel or availability of spare parts may stipulate the need to replace the faulty part by replacing the parent assembly or even entire WRAs in some mission critical applications.

These facts are central to the RBS problem as WS availability hinges on the availability of all the component WRAs. Similarly, a complex WRA may have many SRAs, each with their own SSRAs and so on. Each of these sub-indentured parts all have their own unique failure rates that contribute to the overall failure rate of the WRA. Calculation of availability rates rely in a "rolling-up" of expected backorders (EBOs) through the indenture structure which is further explained in the next section.

2.3 VARI-METRIC Model

At the system level, the problem of optimizing a stocking policy requires the modelling of the failures of each WRA over some time horizon. WRAs can fail because of a failure of any of its sub-indentured candidates so any such model must necessarily consider the failure distribution of each of that WRA's component SRAs. The VARI-METRIC model, developed by F. Michael Slay (1980) and improved by Graves (1985) is an improvement on the Multi-Echelon Technique for Recoverable Item (METRIC) model developed by Sherbrooke (1966). It is described in detail in Sherbrooke's textbook (2004). While METRIC was developed as a system-based model for multiple sites and echelons of supply, Sherbrooke (1971) later showed that the same principles of METRIC are applicable to single site, multi-indenture problems such as the RBS problem.

One problem with the METRIC model is that it does not take variance into account. Rather, it relies on the number of EBOs over the given time horizon based on observed mean

demand, mean repair time and the mean number of units in repair. As might be expected, this method was found to regularly underestimate the number of stockouts, leading to an underestimation of the number of spare parts required, leading to increased WS downtime while awaiting receipt of additional units.

In VARI-METRIC, Slay, Graves and Sherbrooke demonstrate that "a probability distribution whose mean and variance agree with the values for the pipeline will do a lot better than the Poisson" (Sherbrooke 2004). This statement underpins the choice to use the negative-binomial distribution in calculating EBOs, a departure from the Poisson Process used in the METRIC model. Graves (1985) sampled 2304 problem instances and in 227 (11.5%) of cases, the METRIC approximation resulted in an incorrect stocking policy compared to just 18 (0.9%) incorrect decisions made using VARI-METRIC's negative binomial approximation[4]. This is better than a tenfold improvement and provides an adequate basis for its use in the RBS problem described in the next section.

2.4 Readiness Based Sparing

2.4.1 Background

In 2011, the CNO issued OPNAV instruction 4442.5A dictating that, "RBS is to be applied to both aviation and maritime allowance package development, AVCALs, shore-based consolidated allowance lists (SHORCALs), all Marine aviation logistics support packages (MALSPs)," and then goes on to direct the use of A_o as the primary readiness metric, citing OPNAV instruction 3000.12A:, "resource sponsors will establish readiness thresholds in terms of A_o . . ." The Center for Naval Analysis (CNA) first began the process of looking into the use of A_o as the primary readiness metric in 1980 and continued to study and devise implementation strategies until 1987, when the first full scale effort was made by OPNAV to prototype the use of RBS in the DDG-52 program. This initial implementation process was studied and described in detail by Burdick (1991); his illustration of the development timeline is provided in Figure 2.3:

[4]It should be noted that of the 18 incorrect decisions made with the VARI-METRIC approximation, two decisions were wrong in the conservative direction (more stock recommended than needed). No such conservative error was noted using the METRIC approximation.

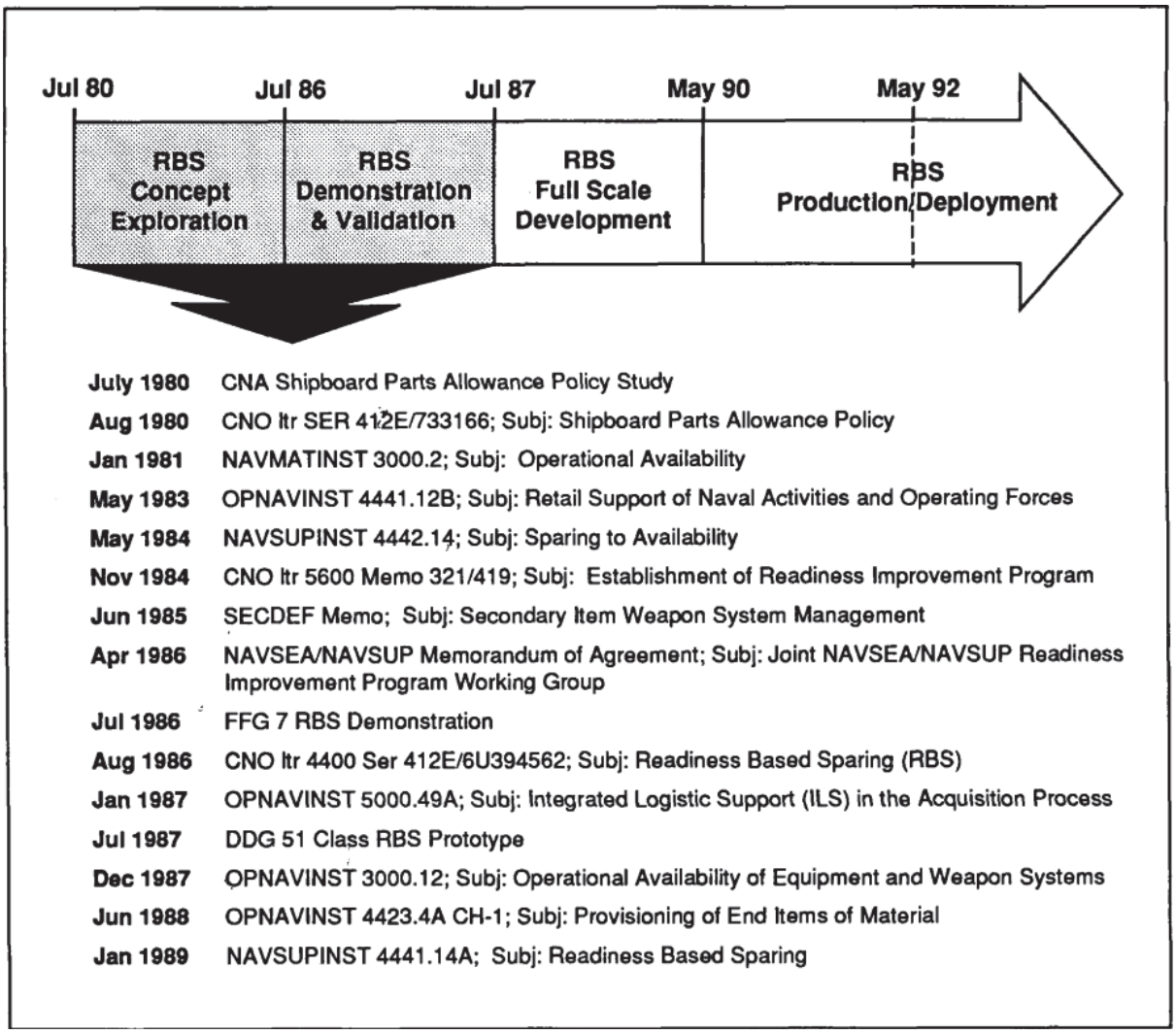

Figure 2.3. Timeline for the study, implementation and policy execution of RBS in the United States Navy (USN).
Source: Burdick (1991).

2.4.2 Readiness Goals, Mission Capable vs Fully Mission Capable

The critical element of the RBS problem is that it looks at an individual type/model/series (TMS) aircraft and takes as input the desired A_o. An optimization algorithm can then use a system level approach such as VARI-METRIC, to determine the expected number of backorders for a given stocking policy which is then used to calculate the expected A_o for

that TMS.

The hitch in the problem from a policy perspective is what constitutes "operationally available." OPNAV defines two different metrics for this: mission capable (MC) and fully mission capable (FMC). According COMNAVAIRFORINST 4790.2D, a MC aircraft is one that "can perform at least one of its missions." The FMC definition is more nuanced, though generally speaking an FMC aircraft is capable of performing all missions for which it was designed. OPNAV stated goals for all aircraft as shown in Table 2.1:

Table 2.1. A_o Goals by TMS. Adapted from OPNAVINST 4442.5a (2011).

	STANDARD		DEPLOYED	
T/M/S	MC GOAL	FMC GOAL	MC GOAL	FMC GOAL
ALL	73	56	78	61

Maintenance planners use TMS specific matrices to determine if a particular equipment failure will result in the maintainers placing the aircraft in a reduced status. To be more specific, an aircraft can be placed in partially mission capable (PMC) status if it is still capable of performing at least one of its missions but not all of them, or in non-mission capable (NMC) status where it is unable to perform any mission.

The difficulty with this system from a pedagogical standpoint is that VARI-METRIC does not care about any in-between status: the system is either functional or not. The MC goal is not considered as there is no current theory to calculate EBOs that apply to one material status (e.g. NMC) and not another (e.g. PMC). Therefore, the current RBS algorithm, described in the following section assumes any component failure results in an NMC aircraft, which conveniently meets the OPNAV requirement that "all [supplemental aviation supply support (SASS)] SASS products will be built to the deployed FMC goal."

2.4.3 The RBS Problem

We discussed the RBS problem has at length by this point, but we have not described it in explicit terms. We can formulate this problem as an optimization problem where the objective function is cost of the stocking policy (C). We have constraints for this problem in the form of required minimum operational availability ($\underline{A_o}(w)$) values for each weapon system (w). Using notation from Salmeron and Buss (2021), we have:

$$\begin{aligned} \min_{S} \quad & C = \sum_{i \in I} c_i s_i \\ \text{s.t.} \quad & A_o(w, S) \geq \underline{A_o}(w) \quad \forall w \in W \\ & \underline{s_i} \leq s_i \leq \bar{s}_i \qquad\qquad \forall i \in I \end{aligned} \tag{2.1}$$

where:

c_i	Unit price of part type i [Dollars]
S	Set of stock levels for all parts types
I	Set of all part types for $i \in I$
s_i	Decision variables, stock level for part type i for $i \in I$
$A_o(w, S)$	Operational Availability for WS w given stocking policy S
$\underline{s_i}$	Minimum stock value for part i, set by policy
$\bar{s}_i$	Maximum stock value for part i, set by policy.

2.5 NAVARM

The current algorithm by which the RBS problem is solved, developed by Salmeron and Buss (2021), is called NAVARM. It was originally developed in 2017 and borrows from Sherbrooke's explanation of the VARI-METRIC model (2004) to calculate EBOs which is then used to calculate A_o using other data provided in site-specific candidate files.

These files are prepared by WSS and provide information needed to construct the indenture structure for each WS, part type data such as demand, and site data such as the quantity and A_o goal for each type of WS at the site.

2.5.1 Expected Backorders

For the RBS problem, we assume a continuous review inventory system operating under an $(s_i, s_i - 1)$ policy, where s_i is target stock level, also called the reordering objective, and $s_i - 1$ represents the reorder point. This means that every time a part fails, a due-in (DI) is created that is either satisfied by issuing stock or else a backorder is created. Since the reorder point is $s_i - 1$, each time a part is issued from stock, a reorder is placed immediately to resolve the stock deficiency. The reason for using an $(s_i, s_i - 1)$ system is that this models the reality of how parts are ordered in the Fleet

The basic equation for calculating backorders is found in Sherbrooke Chapter 2 (2004) and it is the basis for all computations of EBOs in this thesis:

$$\begin{aligned} EBO(S) &= Pr\{DI = s+1\} + 2 \cdot Pr\{DI = s+2\} + 3 \cdot Pr\{DI = s+3\} + \dots \\ &= \sum_{x=s+1}^{\infty} (x - s) Pr\{DI = x\}. \end{aligned} \tag{2.2}$$

Under VARI-METRIC, the number of due-ins at any given time is distributed according to a negative binomial distribution where:

$$Pr\{DI = x\} = \binom{a + x - 1}{x} b^x (1 - b)^a \qquad x = 1, 2, 3, \dots \tag{2.3}$$

Sherbrooke (2004) shows that the parameters a and b in (2.3) can be written in terms of the mean (μ) and the variance-to-mean ratio (VTMR) (V) such that:

$$a = \frac{\mu}{V - 1} \qquad b = \frac{V - 1}{V}. \tag{2.4}$$

In the RBS problem, when calculating the EBO for a particular part type, μ is taken to be the expected "pipeline" (pp) for that part type. Pipeline denotes the quantity of a part type that is either in repair or on order through the supply system. The calculation for pipeline is

simply the demand (m) over the time horizon multiplied by the length of the time horizon[5]. For both NAVARM and our work discussed later we use the following computation for pipeline, as presented by Salmeron and Buss (2021), split into its two components, resupply (2.5) and repair (2.6):

$$pp_{resupply} = \frac{qpa_{w_p} \cdot mrf_p \cdot wfhrs_{w_p}(hpost_p + wdt_P)}{90}, \tag{2.5}$$

$$pp_{repair} = \frac{qpa_{w_p} \cdot rpf_p \cdot wfhrs_{w_p} \cdot imarpt_p}{90}, \tag{2.6}$$

where:

[5]The time horizon used in the RBS problem is one quarter, 90 days, measured in hours (2160 hours).

P	Set of all part types, for $p \in P$
W	Set of all WS types, for $w \in W$
pp_{repair}	Expected pipeline for part type p due to failures resulting in IMF repair[units of the part]
$pp_{resupply}$	Expected pipeline for part type p due to failures resulting complete replacement of part through stock system [units of the part]
qpa_{w_p}	Quantity per application (number of candidates of part type p on WS w) [units of the part]
$wfhrs_{w_p}$	Wartime flying hours in a quarter for WS w that part p belongs to [hours]
mrf_p	Maintenance replacement factor (number of failures per flying hour that are not repairable by the IMF) [failures/hour]
$hpost_p$	High priority order and shipping time [days/failure]
wdt_p	Wholesale delay time [days/failure]
rpf_p	Rotable pool factor (number of failures per flying hour that are repaired at the site) [failures/hour]
$imarpt_p$	IMF repair time [days/failure]
90	Used to convert $wfhrs$ to a daily quantity since $imarpt$, $hpost$ and wdt are given in terms of days.

Once the two components are calculated, they are added together to calculate the total pipeline for the part type:

$$pp_p = pp_{p_{resupply}} + pp_{p_{repair}}. \tag{2.7}$$

where:

pp_p	Expected pipeline for part type p [units of the part].

Equations (2.2) through (2.7) allow us to calculate the number of EBOs for a single part type with a given stocking policy. RBS requires the calculation of EBOs for many part types in a multi-indenture structure, which is generally a challenging problem. For instance, see

the problem shown in 2.2, we will refer to the various candidates by their part types in this figure in the following explanation. In this structure the number of EBOs for WRA A is the sum of the EBOs for D and E, and for E the number of EBOs is the sum of H and I but also J and Q fromWRA C . In this way, the stocking policy for part types J and Q influence the EBO calculation for a WRA (A) that they are not sub-indentured under.

NAVARM deals with this chain of influence structure by building a list of all sub-indentured part types for each part type. The following pseudo code from the NAVARM project documentation (Salmeron and Buss 2021) outlines the basic methodology used to resolve the EBO calculation problem for all part types:

Algorithm 1 NAVARM EBO Rollup Process

Input: Set of all candidates (P), set of all part types (I)
while $\exists p \in P \mid EBO_p = 0$ **do**
    if $ChildrenSolved_p$:= false **then**
        if $EBO_{p'} > 0 \ \forall p' \in P_p^{child}$:= true **then**
            set $ChildrenSolved_p$:= true;
        end if
    end if
    if $ChildrenSolved_p$:= true **then**
        if $ChildrenSolved_{p'}$:= true $\forall p' \in P_{i_p}$ **then**
            set $CommonSolved_p$:=true;
        end if
    end if
    if $ChildrenSolved_p$ = true $\wedge CommonSolved_p$ **then**
        solve EBO_p
    end if
end while
Output: EBO for all $p \in P$

where:

$ChildrenSolved_p$	Boolean that is true once all candidates sub-indentured to candidate p have had their EBO calculated
$CommonSolved_p$	Boolean that is true once all candidates of the same part type as candidate p have had their sub-indentured candidate EBO calculated
EBO_p	Expected backorders for candidate p
P_{i_p}	The set of all candidates of part type i common to candidate p.

The calculation of each individual candidate's EBO uses Equation (2.2) however, Salmeron and Buss use Sherbrooke's METRIC or VARI-METRIC methodology on a case-by-case basis. The number of failures for all candidates are initially modeled using a mean ($\bar{x}_p$) equal to the pipeline for that candidate, calculated using Equation (2.7), and a variance (σ_p^2) also equal to the pipeline for that candidate. Since $\bar{x}_p = \sigma_p^2$ at the beginning of the process, the EBO for all candidates at the lowest level of indenture, leaf nodes, are calculated using METRIC theory: Poisson distributed failures. Once all leaf nodes are calculated, their expected failures are added to that of their parent candidates' part type in the indenture structure. When this "roll-up" process starts, the ratio of variance to mean ($VTMR$) is no longer going to be 1:1. For this reason EBOs are calculated by METRIC theory if $VTMR_i$ part type i is ≤ 1 and by VARI-METRIC theory (negative binomial distributed failures) otherwise.

2.5.2 Operational Availability

The primary constraint in the RBS problem is to ensure that A_o meets some desired minimum value for each TMS in the problem. With the EBOs for each candidate now available, the calculation of A_o is straightforward. The availability of each WRA for a given TMS is assumed to contribute equally to the overall availability for that TMS (Salmeron and Buss 2021):

$$A_o(w, S) = \prod_{p \in P_w^{WRA}} A_o^{WRA}(p, S), \tag{2.8}$$

where:

$A_o(w, S)$	Operational Availability for WS w given stocking policy S
P_w^{WRA}	Subset of candidates in P that are WRAs on WS w
$A_o^{WRA}(p, S)$	Operational Availability for WRA p given stocking policy S.

In order to calculate the availability of each WRA, $A_o^{WRA}(p, S)$, on a given weapon system, we must first calculate the expected number of times the WRA will need to be removed ($Removals_p$) due to a failure:

$$Removals_p = qpa_p \cdot wfhrs_{w_p}(mrf_p + rpf_p). \tag{2.9}$$

We can then calculate availability for WRA p using the NAVARM approximation:

$$A_o^{WRA}(p, S) \approx \frac{1}{1 + \left(\frac{1}{2160} Removals_p \cdot MTTR_{w_p} + EBO(p, S)\right)/N_{w_p}}, \tag{2.10}$$

where:

$MTTR_{w_p}$	Mean time to repair a failure on candidate p's WS w [hours]
N_{w_p}	Number of candidate p's WS at the site [unitless]
$EBO(p, S)$	Expected back orders for candidate p given stocking policy S
2160	Number of hours in a quarter, used to reduce $Removals_p$ which is given in [failures/quarter], to [failures/hour].

The methodology used in NAVARM to calculate EBOs and A_o is used in our MCMC algorithm to the extent that the same Java code base is directly incorporated, with permission of WSS as well as Salmeron and Buss (2021), into the code base for our model.

2.5.3 Marginal Analysis

NAVARM's primary optimization methodology is also borrowed from Sherbrooke (2004) and leverages a heuristic called marginal analysis. This family of heuristics, sometimes called greedy heuristics are defined by Rardin (2019) as "electing the next variable to fix

and its value does the least damage to feasibility and most helps the objective function based on what has already been fixed in the current partial solution."

To accomplish this, Salmeron and Buss create a "shopping list," as Sherbrooke calls it (2004), by first calculating the current EBOs given a minimum stocking policy (setting $S_i := \underline{S}_i$), and the resulting A_o for the weapon system under consideration[6]. NAVARM then calculates the change in EBO and consequently the change in A_o for successive unit increases from $\underline{S}_i$ to $\bar{S}_i$ for each part type $i \in I$. The change in A_o is divided by the unit cost of the part , Salmeron and Buss refer to the result as a "greedy ratio". The "shopping list," another term used in the NAVARM literature, is built using all greedy ratios from all part types and NAVARM selects parts from the top of the shopping list and continues down until $\underline{A}_o$ is met. A more thorough description of this process used by Salmeron and Buss is provided in their NAVARM project documentation and an intuition for marginal analysis as it applies in this setting is found in Sherbrooke Chapter 2 (2004).

2.6 Markov Chain Monte Carlo

2.6.1 The problem with high dimensions

Rardin (2019) describes greedy heuristic algorithms thusly "Much more commonly, they risk suffering from looking only at local information. . . a decision that appears good. . . will actually end up forcing the search into a very poor part of the feasible space." There are numerous examples of greedy heuristics in the literature that produce less than optimal solutions to problems through the entire complexity continuum. That is not to say that NAVARM with its greedy heuristic approach produces necessarily poor solutions; rather it means that we can be confident that the solutions created by NAVARM, while compute resource efficient, are near-optimal at best. With this in mind, we consider other possible methodologies that have been shown to produce near-optimal solutions.

One such method is Monte-Carlo simulation. One could easily conjecture what the result of such an exploratory random approach might have considering the scale of the RBS problem.

[6]NAVARM randomly assigns the order in which WS TMS are considered. The number times it selects a different permutation of this order is set by the user and the best objective function value for each is tracked between permutations and compared after the last to determine the best option.

Many sites considered have upwards of 3,500 part types. Even if there were only 2 possible stock levels to consider, a low estimation, the quantity of different policies that must be evaluated is roughly $4 \cdot 10^{1023}$. Obviously, this implies that, even given all the computing power in the world, one would need an intractably large amount of time even if computing in parallel on a super-computing cluster to examine enough of the decision space to have a decent level of confidence that the best stocking policy found is anywhere near globally optimal.

Another option might be to use a hill climbing method where the algorithm starts at some maximum position, set $S_{0_i} := \bar{S}_i$, and let the algorithm find its way to some minimum value by iteratively sampling all "neighbor" states for a given state and "moving" to the best, feasible neighbor. Rardin (2019) describes this method in detail in chapter 3 of his book. The major problem with a purely exploitive procedure, is that the problem is constrained by A_o goals. As we show in later chapters, there are myriad local optima due to the complex geometry of the feasible region where a given state, or stocking policy, provides adequate A_o for all WS. A purely exploitive hill climbing algorithm will invariably find its way to the "bottom" of some local optima and get "stuck". The only hope for an algorithm like this to succeed would be to stochastically select many feasible starting policies and run many iterations of the hill climbing algorithm, comparing the values of all local optima and choosing the best.

What is proposed is an algorithm that can efficiently sample values from the incredibly complex decision space and move around to explore various regions while also being capable of exploiting regions of increased potential. Ideal candidates for such an algorithm belong a family of algorithms called Markov Chain Monte Carlo (MCMC). First proposed by Metropolis (1953), the refinements needed for our purposes were made by Hastings (1970) that allow us to have an algorithm that can combine the exploratory nature of Monte Carlo simulation and the exploitive nature of a hill climbing algorithm.

The basic principle of the Metropolis-Hastings algorithm is that it moves from state to state in the decision space in a Markov Chain fashion. Hastings describes a method for sampling an h-dimensional distribution by changing one to many elements of the h-dimensional vector. In our application we are not dealing with a probability distribution, but a simpler scenario of optimization in higher dimensions. The principle of using rejection sampling

in the MCMC context is what leads us to the optimization algorithm chosen for this thesis research. Using Hastings's method for determining neighbor points he proposed in 1970, and using Metropolis's acceptance criteria, we have the following Metropolis-Hastings formulation:

Algorithm 2 Metropolis-Hastings applied to the RBS problem

Input: initial stocking policy (S_0), set of all part types (I), desired distribution for ξ, and desired number of steps (k_{max})
$k \leftarrow 0$
while $k \leq k_{max}$ **do**
    for $i \in I$ **do** ▷ Generate random neighbor candidate S'
        $s'_i = s_i + \xi$
    end for
    Calculate probability of acceptance: $P_a = \min\left[1, \frac{Cost|S'}{Cost|S_k}\right]$
    Generate $R \sim Uniform(0, 1)$
    if $R \leq P_A$ **then**
        Set $S_{k+1} := S'$
    else
        Set $S_{k+1} := S_k$
    end if
    $k = k + 1$
end while

Output: $S_{k_{max}}$

where:

ξ	Random integer from some desired distribution
S_k	Stocking policy at step k, for $s_{i_k} \in S_k$
s_{i_k}	Stock level for part type i at step k for $i \in I$
S'	Neighbor candidate stocking policy, $s'_i \in S$‘ $\quad \forall i \in I$.

2.6.2 Simulated Annealing

Kirkpatrick et al. (1983) first proposed using ideas from the natural process of annealing for the optimization of systems as "an adaptive form of the divide and conquer approach." This algorithm is designed to mimic the physical process of annealing. Annealing is the

process of controlled cooling that a heated metal must go through to ensure that internal stresses are dissipated, resulting in a more robust material. The mathematical analog uses the Metropolis-Hastings process but changes the acceptance probability to be a function of another parameter: temperature T. Using the formulation from Saloman, Sibani and Frost (2002) with Hastings's method for determining neighbor points he proposed in 1970, and using Metropolis's acceptance criteria, we have the following Metropolis-Hastings formulation:

Algorithm 3 Simulated Annealing applied to the RBS problem

Input: initial stocking policy (S_0), set of all part types (I), desired distribution for ξ, initial and minimum temperatures (t_0, t_min), and desired cooling schedule ($f(t)$)
$t \leftarrow t_0$
while $t \geq t_{min}$ **do**
    Calculate current temp: $t' = f(t)$
    for $i \in I$ **do** ▷ Generate random neighbor candidate S'
        $s'_{i_t} = s_{i_t} + \xi$
    end for
    Calculate $\Delta E = (Cost|S') - (Cost|S_T)$
    Calculate probability of acceptance: $P_a = \min\left[1, \exp\left(\frac{\Delta E}{T}\right)\right]$
    Generate $R \sim Uniform(0, 1)$
    if $R \leq P_A$ **then**
        Set $S_{t'} := S'$
    else
        Set $S_{t'} := S_t$
    end if
    $t = t'$
end while

Output: S_t

where:

$f(t)$	Desired cooling schedule for the algorithm.
S_t	Stocking policy at temperature t, for $s_{i_t} \in S_t$
s_{i_t}	Stock level for part type i at temperature t for $i \in I$.

This algorithm allows for a significant amount of freedom to:

1. Select the desired move class (ξ), and even tailor the selection criteria based on any number of desired algorithm parameters or state variables.
2. Control the cooling schedule ($f(t)$) and even re-heat based on desired criteria.
3. Set selection and acceptance criteria as tailored to the problem structure as desired.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

The MCMC algorithms presented at the end of Chapter 2 are straightforward and concise, however the RBS problem is anything but concise. The RBS constraints, stated in (2.1), mean that any MCMC algorithm must account for feasibility of every point considered. While a simulated annealing algorithm is relatively simple to write, the bulk of the effort for this project has been to resolve how to manage the complexity of the A_o constraint. In order to adequately frame the processes designed to meet this challenge, we first discuss the data and the basic flow from input to output.

3.1 Data Flow

The data for the aviation RBS problem is generated in the fleet and collected, cleaned, collated and maintained by NAVSUP WSS in Philadelphia, Pennsylvania. This data is then parsed into site specific database files, called candidate files. While a candidate file has much more data than is used in the RBS problem, the data used can be separated into several categories:

1. Site related data
    - Enumeration of the various TMS
    - Repair capability
2. WS data
    - Flying hours
    - A_o Goal
    - Repair time
3. Indenture structure
    - WRA / SRA indenture level
    - Parent candidates and part types
4. Part data
    - Cost
    - Failure rates

- Requisition lead times
- Repair data

The data, once compiled into the candidate file can be passed to various applications.

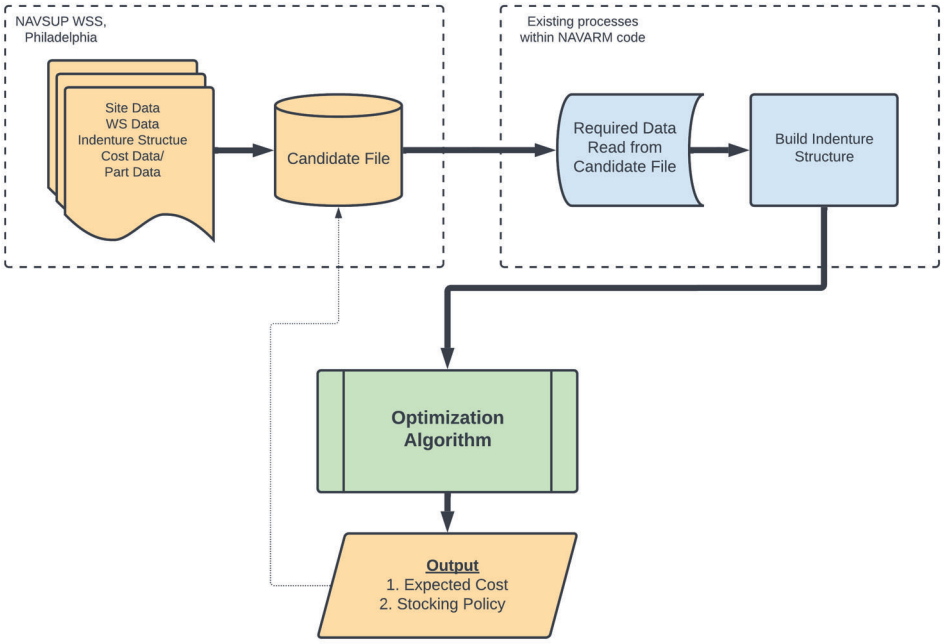

Figure 3.1. Macro view of the flow of data from source input to output of solution.

In order to use this data in an optimization algorithm, there is a significant amount of pre-processing is required . Reading the data from the database file is the first step in this process and requires the interfacing software to query many tables to obtain the required data. This data must be read and organized in local memory in such a way that it can be efficiently accessed when building the indenture structure. Objects must be created for each TMS, part type, and candidate. These objects are then given attributes that allow the indenture structure to be interpretable whether considering a single candidate, a whole part type or an entire WS.

Since NAVARM (Salmeron and Buss 2021) already performs these data collating and indenture building functions, it was desirable to obtain the codebase and leverage this

functionality. The Java codebase, obtained from WSS with permission of Salmeron and Buss, was selected over its Visual Basic for Applications (VBA) version due to the more robust development environment that is available with Java as compared to VBA. Once all unnecessary code was removed from what was needed to produce the required structure, we were able to start scaling the problem.

3.2 Problem Scaling

In scaling the NSAM algorithm, we divide the development into three phases:

- Phase 1: A simple, knapsack-type, constant A_o
- Phase 2: A problem with multiple WRA, limiting probability derived A_o
- Phase 3: Multi-indenture RBS Problems

By following this strategy, we see patterns emerge at smaller scale and less complexity that allow for design choices to be made in an iterative approach. Each step offers a chance to explore the RBS problem structure from different angles, and produced a robust application that is exceptionally modular and capable.

3.2.1 Knapsack-type, constant A_o

In this entry-level problem, the goal is to create a simple simulated annealing algorithm in the Java environment. At this early stage, avoiding the need to bring in the complexities that come with RBS allow a focus on proficiency with the language and building blocks required for any similar algorithm. The following assumptions are made for this step of the process:

- Each part in the problem contributes a constant A_o and cost. This bears some definition since it is a departure from the intuition of the actual RBS problem discussed in Chapter 2. In the classic knapsack problem, every item to be put into the bag has a given size and weight, while the bag has a known size capacity. Likewise in this iteration we consider parts with known cost and a specified contribution to the overall A_o. This is not unlike the concept behind Salmeron and Buss's (2021) greedy ratio however, there is no EBO calculation so the A_o value is completely arbitrary.

- There is no indenture structure. Unlike the RBS problem, the introductory level of our algorithm does not consider the inherent "engineering parts hierarchy" (Sherbrooke 2004) of a multi-indenture problem. This is done to accommodate the A_o simplification described above.
- Minimum stocking quantities for all part types is set to zero: $\underline{S_i} = 0 \; \forall \; i \in I$
- Maximum stock levels are all determined stochastically based on assigned cost

This iteration of our algorithm randomly generates n parts and assigns each of them a unique random number that is used to create realistically proportioned approximations of:

- Cost
- A_o Contribution
- Maximum stock level

Dealing with Feasibility

To optimize this problem structure we first must select initial system temperature and a cooling schedule. It becomes clear at this point that we must address the prime complexity of approaching the RBS problem with this family of algorithms: What do we do when a step is found to be infeasible? In other words, is A_o for S' less than the minimum A_o acceptable for the problem? In this iteration, it was determined that the most efficient way to deal with it was to include another term when calculating the acceptance probability: ΔA_o, the change in A_o from the current policy, S, to the neighbor policy, S'. Starting at step 8 in Algorithm 3 where the change in energy, ΔE, is calculated we amend as follows:

Algorithm 4 Calculation of P_a for the knapsack-type treatment of the RBS Problem

Input: S',S, $A_o|S'$
Calculate $\Delta E = (Cost|S') - (Cost|S_T)$
Assign Boolean control variables:
if $\Delta E \geq 0$ **then** ▷ We want to reward reduced cost
    $E_{bool} = 0$
else
    $E_{bool} = 1$
end if
if $(A_o|S') \leq \underline{A_o}$ **then** ▷ We want to reward feasible A_o
    $A_{bool} = 0$
else
    $A_{bool} = 1$
end if
Calculate probability of acceptance,P_a
$P_a = \min\left[1, E_{bool} \cdot \exp\left(\frac{\Delta E}{T}\right) + A_{bool} \cdot \exp\left(\frac{\Delta A_o}{T}\right)\right]$
Output: P_a

This treatment of the Cost - A_o trade-off resembles a Lagrangian Relaxation and in fact another variable, the Lagrange multiplier, could very easily be added to relax A_o's contribution to the acceptance probability calculation.

Temperature Limits

We have described a method to account for A_o that rewards feasible A_o with increased probability of acceptance, and vice versa, however it was also decided to allow the algorithm to enter the infeasible region for a set number of states. This allows the annealing algorithm to potentially find otherwise unreachable states while not letting it wander endlessly through infeasible space and eventually get stuck when the temperature gets too low to allow infeasible moves with any realistic probability[7]. We do this with logic that stores in memory

[7]With this simulated annealing method, there is theoretically never a probability of zero for any state transition regardless of cost or feasibility; however, given the limits of computational precision, below a certain probability (different based on the machine or code) it is effectively a zero probability.

the last feasible solution while counting the number of states until reaching the maximum number of allowable infeasible steps. If no feasible solution is found, then the algorithm reverts back to the last feasible solution and continues. If a feasible solution is found then the it is accepted using the same acceptance logic described in Algorithm 4.

Next we must select our initial temperature, T_0 and final temperature T_{min}. At this stage in development we set an arbitrarily high T_0 and an arbitrarily low T_{min}, but we will come back to a discussion of selecting T_0 in 3.3.

Cooling Schedule

The other temperature related decision to be made is in selection of a cooling schedule. This is how the system changes temperature from T_0 to T_{min}. There is much literature on schedule selection, however at this point in the process, we elect for a simple linear cooling rate where the next temperature is calculated as the product of the current temperature and some constant very near 1.

Move class

Lastly we develop the mechanism by which neighbor states are selected, the Move class. To build each neighbor state, S', the Move class must randomly select a new value for every part type in the policy. There are a few considerations that must be made when devising a methodology for this:

- **Scale**. Factors such as failure rate, unit of issue, and cost require that each part type be considered individually when drawing its next move from a distribution. For instance, a site may require dozens of a particular valve cap, but seldom will require more than a handful of a particular circuit card, so moves for these two parts must be drawn from a distribution scaled individually.
- **Shape**. While it may be desirable to have a symmetric distribution such as the Gaussian that absolves us of the need to decide on a shape, such a distribution prevents us from being able to "shepherd" the algorithm one way if desired.

These considerations lead us very naturally to a Weibull distribution with a scale parameter

(α) based on total demand for the part type and a shape parameter (γ) based on user input[8]. However, since this distribution is continuous, we approximate it by using a ceiling function[9]. Consideration must be given to a third parameter, location (μ), when choosing this Weibull approximation since without some correction, all values selected from the distribution will be positive. To solve this, we devise a correction that always shifts the distribution such that the mode of the distribution is always approximately zero. Our approximation is:

$$f(x) = \left\lceil \frac{\gamma}{\alpha}\left(\frac{x-\mu}{\alpha}\right)^{(\gamma-1)} \exp\left(-\left(\frac{x-\mu}{\alpha}\right)^{\gamma}\right) \right\rceil, \tag{3.1}$$

with the mode correction equation:

$$\mu = -\alpha * \left(\frac{\gamma-1}{\gamma}\right)^{\frac{1}{\gamma}}, \tag{3.2}$$

where:

$f(x)$	Mode corrected Weibull approximation
α	Scale parameter set by user
γ	Shape parameter taken to be the square root of the part type's demand over time horizon
μ	Location Parameter as calculated.

Using the inverse transform method, we are able to generate pseudo-random numbers from our distribution. Example output from just such a simulation is provided in Figure 3.2.

[8]Consideration was given to making these parameters temperature dependent, however this was never implemented due to time constraints

[9]Ceiling is used versus simple rounding. Since not all part types are chosen, as discussed later, we want to ensure part types that are chosen actually "move". The use of the ceiling function prevents a value of zero from being selected.

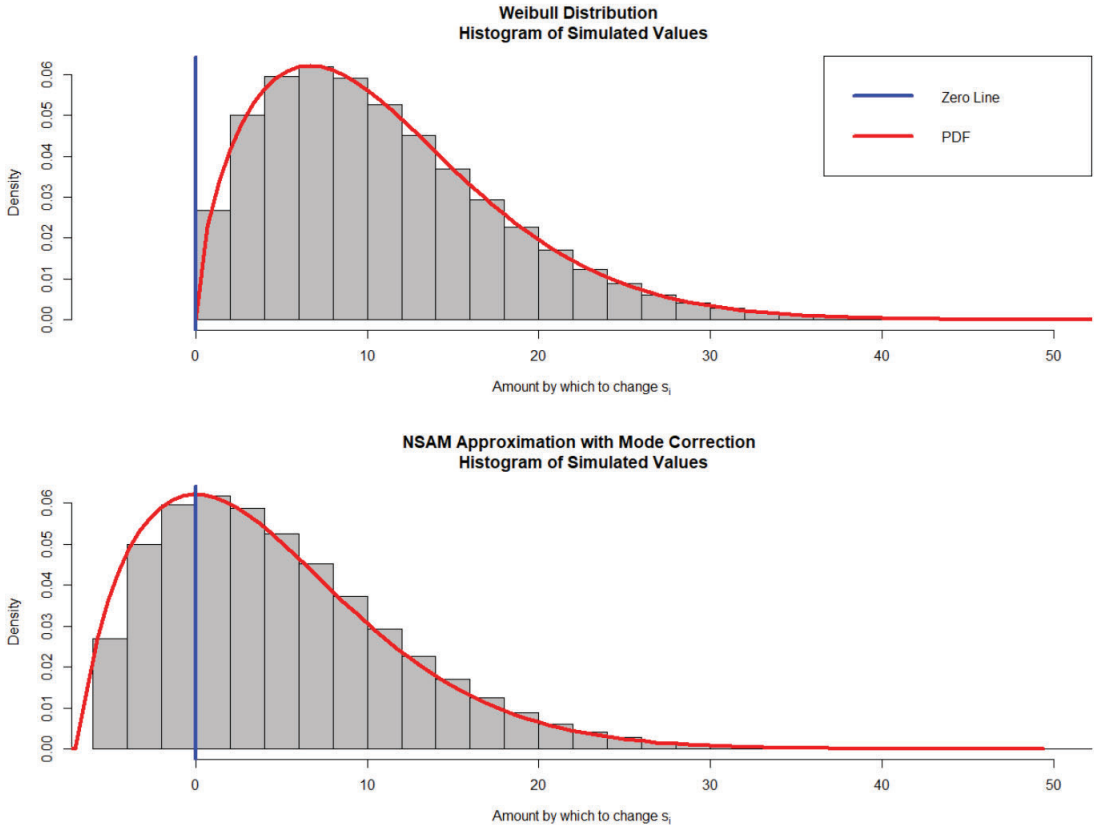

Figure 3.2. Plots of simulated values for both a Weibull distribution with shape parameter of 1.6 and scale of $\sqrt{150}$, and the NSAM Mode-corrected Weibull Approximation with the same parameters. Corresponding density functions are overlaid in red.

With this Move class we can tailor the shape and scale as needed for each part type and keep the most common changes close to zero. It is not always desirable for the distribution to be right tailed as it had been described so far. The other utility of the Weibull discussed was that we could use it to "shepherd" the algorithm in some particular direction. While there can be other reasons this would be done, the particular instance used in our methodology is predicated on the assumption that near-global optima for the RBS problem will have no slack in the availability constraints. As such, shepherding the algorithm to stay space near to the feasible boundary may be desirable. To accomplish this, we add a Boolean multiplier, τ to line 6 of our simulated annealing algorithm (Algorithm 3 from Chapter 2), such that:

$$s_i' = s_i + \tau \cdot \xi \quad \forall i \in I, \tag{3.3}$$

where:

ξ	Random integer simulated using NSAM Mode-corrected Weibull Approximation
τ	Shepherding variable to control whether the distribution is left (-1) or right tailed (1) based on feasibility.

This effectively reflects the NSAM distribution across the line $x = 0$ which allows us to control which direction the algorithm tends towards. If a more symmetric distribution is desired, the shape parameter can be increased ($\gamma = 3.5$ produces a near symmetric distribution) and as the distribution becomes more symmetric, the effects of τ are negated.

Performance

Before adding stochasticity and other layers of complexity, we validate functionality at this point. At this level of complexity, both objective function and constraints are linear with integer variables so we are able to test in Python Pyomo, Bynum et al. (2021) and Hart et al. (2011), using the Gurobi solver (Gurobi Optimization, LLC 2022). Using several iterations with various quantities randomly generated part types, we were able to show that the NSAM solution matched the Pyomo solution each time. This is the only opportunity we were able to truly validate performance with certainty. As we add more complexity, our testing is only performed by comparing results with NAVARM.

3.2.2 Simple indenture, limiting probability derived A_o

At this step we add stochastic part failures and a very rudimentary indenture structure. Additionally we no longer assume given values for maximum stock levels but calculate them using a Poisson Distribution such that:

$$\begin{aligned} \min_{\bar{S}} \quad & \bar{S}_i && \forall i \in I \\ \text{s.t.} \quad & F(\bar{S}_i) \geq 0.9 && \forall i \in I \\ & \bar{s_i} \geq \underline{s_i} && \forall i \in I, \end{aligned} \tag{3.4}$$

where:

$F(x)$ The cumulative density function (CDF) of a Poisson distribution with rate parameter equal to the pipeline, PP_i, calculated using (2.7).

Limiting Distribution Method for Calculating A_0

At this level of complexity, we only consider a system with two WS, each with two WRA arranged as shown in 3.3

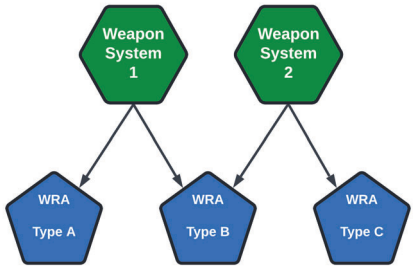

Figure 3.3. A simple problem with two WS each with two WRA where one WRA is common to both WS.

With such a simple system, it is possible to devise an analogous birth-death process that uses the properties of continuous time Markov chains (CTMCs) to determine A_0. We already know that A_0 is the proportion of time that a WS is capable of executing missions. In the RBS setting, this is equivalent to the proportion of time that all parts are functional. Since we assume there is no repair down time if a failed part has a replacement on hand, and that inter-failure time is exponentially distributed, we can build a Markov chain to represent the state space of possible states visited by one of the WS in our small sample problem if the stocking objective (s_i) for both part types on the WS is two ($s_A = s_b = 2$):

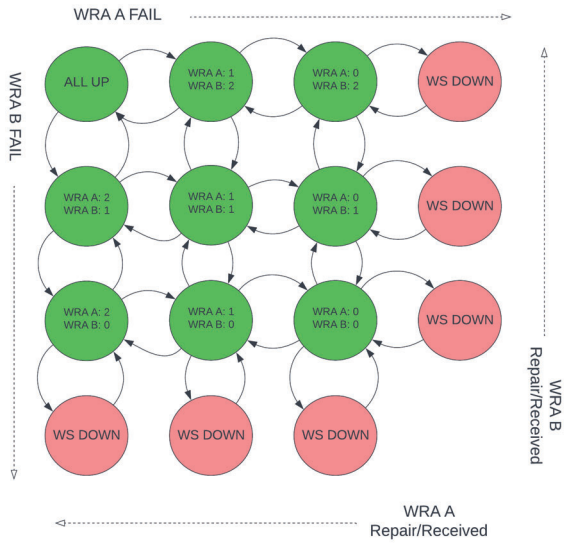

Figure 3.4. Markov Chain representing the possible WS readiness states. The numeric quantities reflect the quantity of parts on hand, ready for issue, of the indicated type. In this example, $s_A = s_b = 2$ meaning only the WS will go down if both parts are issued and the last part issued fails on the WS.

The readiness of both WS in our small problem represented by Figure 3.3 can be represented with similar CTMCs to Figure 3.4. Since we assume parts fail independently, we can represent the quantity on-hand for each part as its own CTMC where the state variable (ρ_i) is defined on the interval $0, 1, \ldots, s_i + qty_{ws_i}$ where qty_{ws_i} is the number of WS at the site that use part type i.

To obtain the rates for the part type i's infinitesimal generator matrix (Q_i) we consider the rate of failure (λ_i) to be the number of removals in a day for a given part ($removals_p$). We calculate this using the result given by (2.9), which returns values in units of removals per quarter, and dividing by 90 (the number of days in a quarter). The rate of failure remains constant until there are no more ready for issue parts and WS begin to go down since we assume a downed aircraft cannot generate additional part failures. Once the number of functioning WS starts to drop, the rate drops in a linear fashion until there are no more functioning WS and the λ is zero.

Calculating repair and replace rates (μ_i) takes a little more work and another assumption. For this CTMC model to be valid, we must assume that parts are repaired or replaced with inter-arrival times that are also exponentially distributed. Since μ_i is the resultant rate of two separate processes, IMF repair and supply system requisitions, we use the following equation to calculate μ:

$$\frac{1}{(hpost_i + wdt_i) \cdot \frac{mrf_i}{mrf_i+rpf_i} + imarpt_i \cdot \frac{rpf_i}{mrf_i+rpf_i}}. \tag{3.5}$$

The denominator is a calculation of the expected repair or replace time. We must assume each failure is a probabilistic combination of the probability that the failure can be repaired at the IMF, $rpf_i/mrf_i + rpf_i$, which takes $imarpt_i$ days, and the probability that the failure must be replaced from the Navy stock system, $mrf_i/mrf_i + rpf_i$ which takes $hpost_i + wdt_i$ days. Since this calculation results in an expected time, to obtain the rate, μ, we simply take the reciprocal of the expected time (since we have already assumed exponentially distributed inter arrival times). The resulting infinitesimal generator matrix has the form:

$$\mathbf{Q_i} = \begin{array}{c} \begin{matrix} \rho{=}s_i + qty_{ws_i} & s_i + qty_{ws_i} - 1 & s_i + qty_{ws_i} - 2 & s_i + qty_{ws_i} - 3 & \cdots & \text{qty}_{ws_i} - 1 & \text{qty}_{ws_i} - 2 & \cdots \end{matrix} \\ \begin{bmatrix} -qty_{ws_i} & qty_{ws_i} & 0 & 0 & \cdots & 0 & 0 & \cdots \\ \mu & -(\mu + qty_{ws_i}\lambda) & qty_{ws_i} & 0 & & & & \\ 0 & 2\mu & -(2\mu + qty_{ws_i}\lambda) & qty_{ws_i}\lambda & & & & \\ \vdots & & & \ddots & & & & \\ 0 & & & s_i\mu & -(s_i\mu + qty_{ws_i}\lambda) & qty_{ws_i}\lambda & & \\ 0 & & & & (s_i+1)\mu & -((s_i+1)\mu + (qty_{ws_i}-1)\lambda) & (qty_{ws_i}-1)\lambda & \\ \vdots & & & & & & \ddots & \\ 0 & & & & & & (s_i + qty_{ws_i})\mu & -((s_i + qty_{ws_i})\mu) \end{bmatrix} \end{array}$$

Figure 3.5. Infinitesimal generator matrix for part i.

With this matrix created, we can use the equations described by Ross (Ross 1997) [10] to produce a vector of the limiting probabilities for part type i, L^i. Since we are measuring availability, we assume 100% availability in any state where $\rho \geq s_i$, but we must correct for reductions in qty_{ws_i} as ρ increases. A general equation to calculate A_o from L^i for a problem with one level of indenture such as we have at this point:

[10]Using equation (6.20) from section 6.5

$$A_o(i,S) = \sum_{\rho=0}^{s_i^p + qty_{ws_i}} L_\rho^i \frac{\min[qty_{ws_i}, \rho]}{qty_{ws}}, \tag{3.6}$$

where:

$A_o(i,S)$	Operational availability of part type i given stocking policy S
P_w^{WRA}	Subset of candidates in P that are WRAs on WS w
L_ρ^i	The limiting probability of state ρ for part type i
qty_{ws_i}	Quantity of WS that use part type i.

Once A_o is calculated for all part types, we multiply all together as in (2.8) since there we are only considering one level of indenture at this point. This method for calculating A_o produces works quickly at the scale we need; however, as the problem ramps up in complexity, more and more matrix operations will need to be completed. A possible method to calculate availability using this methodology at full RBS problem scale is proposed in Algorithm 5.

Algorithm 5 Calculation of A_o using Limiting Probabilities

Input: Baseline A_o^i for all part types using 3.6, indenture structure
$A_oSolved_p$ = FALSE $\forall p \in P$
while $\exists p \in P \mid A_oSolved_p$ = FALSE **do**
    if then $P_p^{child} = \emptyset$
        set $A_o^p := A_{o_p}^i$
        set $ChildrenSolved_p$:= true;
        set $A_oSolved_p$:= true;
    end if
    if $ChildrenSolved_p$:= false **then**
        if $A_oSolved_p$ = true $\forall p' \in P_p^{child}$:= true **then**
            set $ChildrenSolved_p$:= true;
        end if
    end if
    if $ChildrenSolved_p$:= true **then**
        set $A_o^p := A_{o_p}^i \cdot \prod_{p' \in P_p^{child}} A_o^{p'}$;
        set $A_oSolved_p$:= true;
    end if
end while
Output: Vector of A_o by WS

where:

A_o^i	Baseline operational availability for part type i calculated using 3.6
A_o^p	Operational availability for candidate p
$A_{o_p}^i$	Baseline operational availability for candidate p's part type
P_p^{child}	Subset of all candidates for candidate p
$A_oSolved_p$	Boolean that is 1 once the A_o has been calculated for candidate p
$ChildrenSolved_p$	Boolean that is 1 once A_o has been calculated for all $p' \in P_p^{child}$.

While the need for many matrix operations may tax most CPU's, modern graphic processing units have architecture purpose built for these kinds of operations and would likely reduce

time required to process a full scale RBS problem.

Run-time

Using this method, with three WRAs parts, we are able to run 100 MCMC chains of 1600 steps, in series, in under 30 seconds using an Intel Core I9-9900k. It is not known how adding complexity such as more levels of indenture will impact run times, however complexity of the current algorithm is O(n) and implementations of Algorithm 5 are likely to be on the order of O(n^2).

3.3 NAVARM Simulated Annealing Model

With the basis of our simulated annealing model in place, we move to the full scale RBS problem. The obvious change is the addition of multiple layers of indenture under multiple WS. To address this challenge we elect to use multiple classes from NAVARM's codebase to lighten the development load. There are three versions of this final algorithm, affectionately named Vanilla, Chocolate and Strawberry. The algorithms developed in the preceding sections are carried forward in these versions with the exception of the A_o calculation which is discussed in the following section.

3.3.1 Vanilla

Vanilla is the first attempt to optimize full scale RBS problem. Named for its elementary approach to simulated annealing, Vanilla is a bare bones simulated annealing algorithm taken directly from the Saloman et al 2002 chapter labeled "Bare-Bones Simulated Annealing." Our only deviation from the base simulated annealing algorithm is that we carry forward the invisibility logic discussed previously.

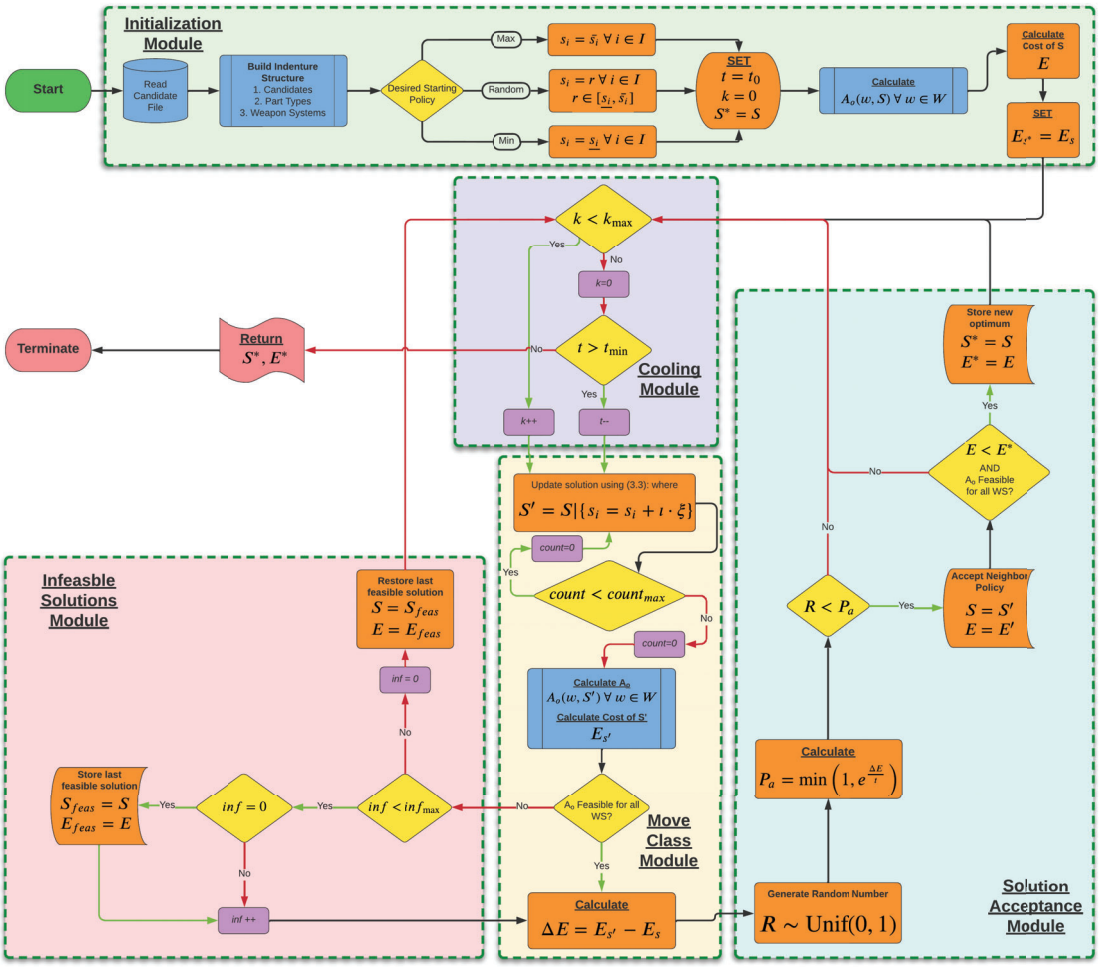

Figure 3.6. Flow chart for NSAM Vanilla. Items in blue are processes performed using code taken directly from NAVARM Java.

Initialization Module

Vanilla takes two input files. First is a JSON [11] file containing all desired NSAM input parameters. The second input is the candidate file supplied by WSS. Once read, the data is used to create the elements of the indenture structure necessary to calculate A_o for each WS. Based input from the JSON file, an initial stocking policy is built that can be any of the following:

[11] JavaScript Object Notation, commonly used to pass large quantities of parameters into Java applications

- Maximum allowance. This option sets the allowance for all part types to their maximum, $\bar{s_i}$. Maximums can be set manually for a part type in the configuration file, but more commonly it is determined by calculating the value of the 0.99 quantile of a Poisson distribution representing number of failures[12].
- Minimum allowance. Sets the allowance for all part types to their minimum, $\underline{s_i}$. As with max values minimums can be set manually, but again it is more commonly determined by calculating the value of the 0.5 quantile of a Poisson distribution[13].
- Random Allowance. Randomly assigns a value, between the minimum and maximum, to each part type in the policy.

The initial A_o is then calculated and all required variables are initialized.

Cooling Module

Once initialized, the algorithm generates, evaluates and accepts (or rejects) policies according to a user defined cooling schedule. At each temperature, NSAM generates k policies to be evaluated. Typical values for k from the literature range from 10 to 100 to get an adequate assessment of the "neighborhood" before reducing the temperature.

Our methodology for selecting T_0 came from Saloman et al. 2002 who proposed a simple sampling method to determine the scale of possible ΔE values for the problem given other algorithm which are discussed later. Taking this and using the standard equation for P_a from Algorithm 3, we set a desired acceptance probability to begin the algorithm. White (1984) recommends values of T_0 that allow for high variability in the beginning.

We experiment using different values for T_0 as well as different values for k, T_{min}, number of steps, and various cooling schedules. results of this experimentation are further discussed in the next chapter.

In practice, the cooling module takes as an input the desired number of temperature states. This is then passed through the cooling schedule function to determine the next value for temperature. For discussion, we simplify this concept to the idea of using a minimum temperature, versus using a counter for the total number of transitions. This is a generally

[12] 0.99 is a typical value. This is a setting that can be changed in the candidate file

[13] Again, this is a typical setting that can be changed

practiced method in the literature and it allows for control over how rapidly the algorithm steps through the temperature profile defined by the cooling schedule.

Move class Module

For every k at every t, the Move class module creates a neighbor policy (S') based on the currently accepted policy (S). To do this, m part types are selected from S (m is defined by the user), then applies the Mode Corrected Weibull approximation, 3.1 through 3.3 where the shape parameter is defined by the user and the scale parameter is calculated as the square root of $\bar{s}_i$.

S' is then evaluated to determine the cost, E', and A_o for each WS. Finally a determination is made to label S' as feasible or infeasible, and ΔE is calculated by comparing to the cost, E, of S.

Infeasible Solutions Module

If S' is infeasible, it is not rejected immediately. NSAM counts the number of infeasible solutions that have been generated before accepting or rejecting. The decision to place this counter before accepting a solution was arbitrary; it is likely better to count only those infeasible policies that are accepted. As long as this counter is less than some user defined maximum value, $inf < inf_{\max}$, the solution will be labeled as an allowed infeasible solution. Once $inf = inf_{\max}$ then the algorithm resets to last solution that was determined to be feasible, S_{feas}.

Solution Acceptance Module

Feasible and allowed infeasible solutions are accepted or rejected using the basic simulated annealing logic from Algorithm 3. If a solution is rejected the algorithm returns directly to the cooling module to begin the next iteration. If it is accepted then the current policy is updated, $S' = S$. Additionally, if the solution is feasible, the newly updated solution is compared to the current best policy: if $E < E^*$ where E^* is cost of the current best solution, then S^* is updated.

3.3.2 NSAM Chocolate

As we will discuss in detail in Chapter 4, results from experimentation with Vanilla indicated that the structure of the RBS problem works against basic annealing methods. Chocolate is proposed to exploit what we know about the problem itself. The primary issue for Vanilla is that letting the algorithm select parts completely at random means that complex and expensive WRAs and SRAs with sub-indentured parts will be selected and removed from the policy. When this happens, it follows that an increase in EBO will occur, and an many cases the removal of even one of these parts from the policy will result in an infeasible solution.

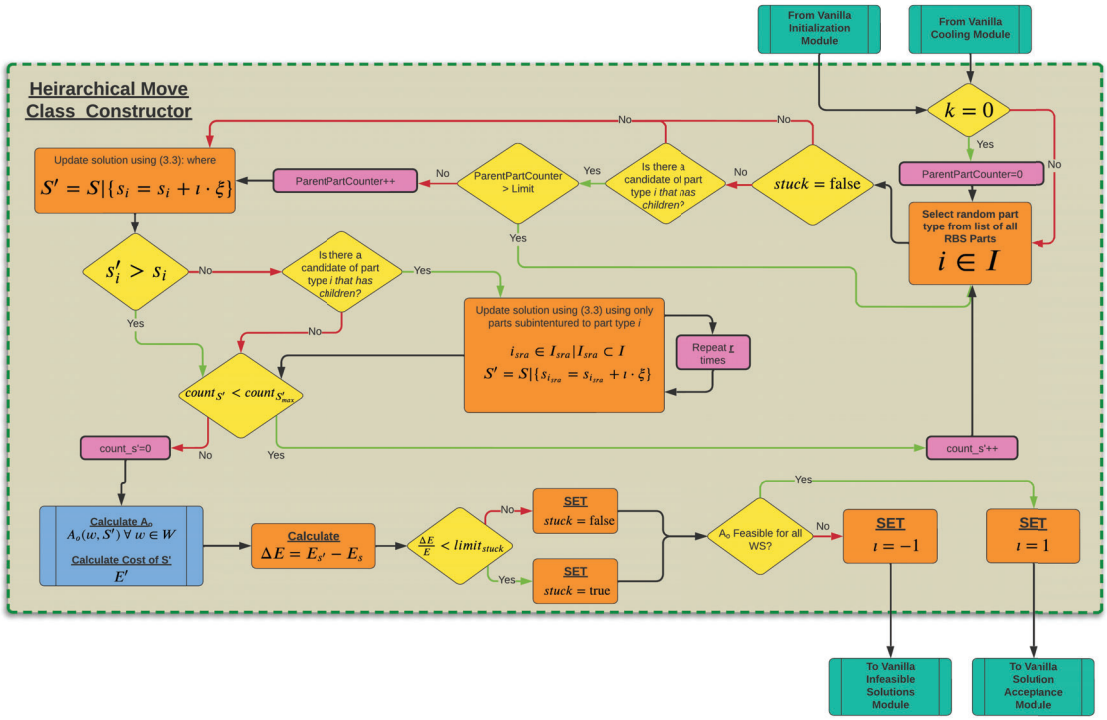

Figure 3.7. Flow chart for NSAM Chocolate.

Chocolate seeks to fix this by completely swapping out Vanilla's Move class module for one that prioritizes parts with less complexity in two ways:

- Selection. When selecting parts for S', a counter is implemented that imposes a hard limit on sub-indentured parts. This counter is reset immediately before building S'.

Once this limit has been reached, every time a part type with children is selected, it is put back in the pile and a new part type is selected. This repeats until the requisite quantity of part types have been selected for change in the given step, $count = count_{max}$.

- Sub-indentured part boosting. If a part type is selected and its quantity is reduced, $s'_i < s_i$, then the algorithm will take a number, r, of the part types that are sub-indentured, i_{sra} to the originally selected part type i and increase their stocking quantities within S'. In our algorithm, r is taken to be a random uniformly distributed integer between 0 and the total number of indentured parts to part type i.

With this added logic, it was noted that when building initial policies from minimum values, the algorithm would get "stuck" when the parent part counter would reach its limit and addition of simple, non-parent type parts would reach the maximum stock limits for those parts. To correct this, logic is implemented that monitors the change in cost, ΔE, from step to step and triggers Boolean variable, labeled $stuck$ in 3.7, when ΔE falls below a user defined proportion of the current cost for a set number of steps. Once triggered, this Boolean causes the algorithm to skip the parent part logic and allow parent part types to be selected even if the parent part counter has reached its limit.

3.3.3 NSAM Strawberry

The final iteration of NSAM, version Strawberry, was created when patterns in Chocolate's initial solution building were analyzed. With some exception, which will be discussed in Chapter 4, to this point in the development process, the efficacy of marginal analysis in the RBS application has been supported by our results. In both the base annealing algorithm, Vanilla, and the boosted, Chocolate, algorithms there is still a large gap between NAVARM and NSAM solutions.

Strawberry borrows heavily from NAVARM, using all the modules needed to perform marginal analysis by creating the greedy ratios discussed in Chapter 2. The underlying principle behind this version is an epsilon-greedy approach to building the initial policy from where the simulated annealing algorithm can depart. Epsilon-greedy has roots in reinforcement learning, however we simply use it to find a starting solution that is random, but in the same neighborhood as the NAVARM solution. For this we return the Move

class to its vanilla form and slightly amend the initialization module, other modules remain unchanged in this version.

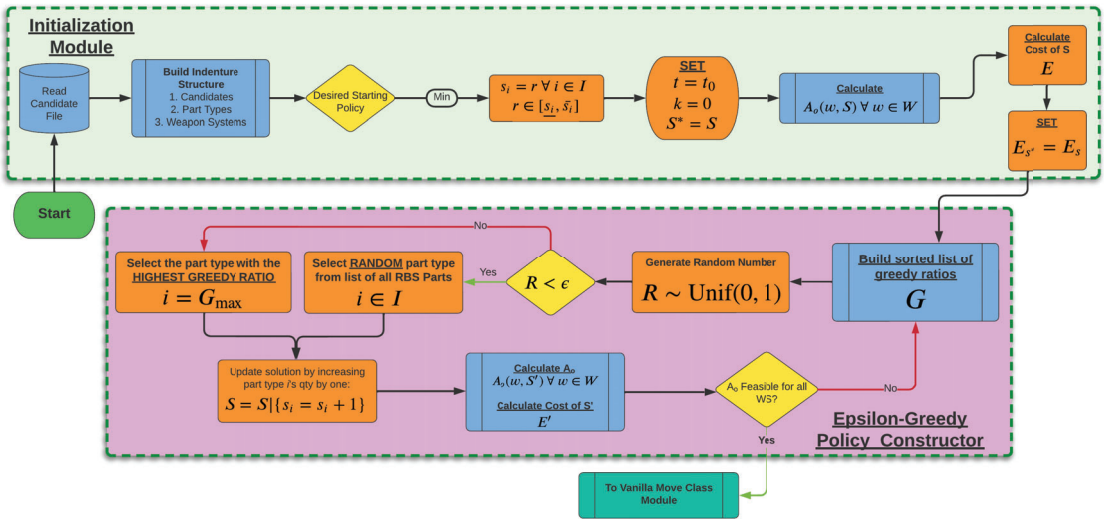

Figure 3.8. Flow chart for NSAM Strawberry.

Strawberry requires starting from minimum values, $s_i = \underline{s_i} \forall i \in I$. Then, using NAVARM's methodology, greedy ratios are calculated for all values of all part types and then sorted from highest to lowest, G. We then generate a uniform random variable on the interval $[0, 1]$, R, and if that value is less than the user defined value for ϵ, we take the first greedy ratio off the top of the list, G, and increase this part type's value by 1. A_o is calculated at this point and if feasible this initial solution is passed to the simulated annealing portion of the algorithm; if not, greedy ratios are re-calculated and the cycle repeats until the solution is feasible.

The version of Strawberry we have described so far was primarily designed for use with a single WS problem (this problem is further discussed in the next chapter). The most advanced version of our attempt to leverage epsilon-greedy policy building incorporated logic that allowed for multiple WS types to allow for full scale testing. The problem created by our approach so far and introducing multiple WS is the calculation of greedy ratios. With one WS there is no concern for the order in which WS are considered; however, with multiple WS we must apply logic to determine this order. As in NAVARM, the order in

which WS are considered when calculating these ratios plays a large part in the order in which parts are added to a policy when using marginal analysis. This is because addition of part types, even when shared between multiple WS types, lead to different changes in EBO for each WS so the order parts selected to add will not be the same.

The logic we devise for this version of our algorithm evaluates A_o for all WS every time a part is added to the initial policy and recalculates greedy ratios at any point where the difference between A_o of the current primary WS[14] and that of the WS with the lowest A_o reaches a set limit. For our experimentation we set this to 0.03. In addition to evaluating at this set point, the same evaluation is performed any time a Metropolis move is made (when $R > \epsilon$).

[14]What is meant by "primary WS" is the WS that is considered first when the current set of greedy ratios were calculated

CHAPTER 4:
Results and Analysis

Experimentation was performed using each iteration of the NSAM algorithm. The purpose behind this experimentation was to try and tune the input parameters needed for simulated annealing. Each run was compared to a NAVARM solution, although there are various settings within NAVARM that can be used to refine and polish solutions, unless otherwise noted, results obtained from NAVARM are unrefined and unpolished. Generally speaking, NAVARM's marginal analysis performed significantly better than NSAM in all but a few cases that will be presented. This chapter will discuss results of testing in three separate cases:

- "Toy" Problem. A simple problem with two WS, each with identical indenture structures consisting of three levels of indenture and only 14 part types.
- Small RBS site with one type of WS with multiple WRAs and three levels of indenture and 741 part types.
- Full Scale RBS problem. Complex CVN site with seven different WS and 4,804 part types across 4 levels of indenture.

4.1 Toy Problem

This problem is the only one that we are able to consistently demonstrate that an MCMC approach offers promise in improving upon the incumbent method, NAVARM, which uses marginal analysis. For this problem, the only algorithm used in testing was NSAM Vanilla since the other two, Chocolate and Strawberry, were designed specifically to deal with complex indenture structure that simply does not exist in the toy problem.

4.1.1 Parameter Tuning

In tuning our model to optimize performance for this problem, we use a factorial experiment with a few values selected for each of the following NSAM parameters:

- Move class Weibull approximation shape parameter 1.2, 3, 6

- Number of interstate transitions, K 5, 10, 15, 20
- Number of part types to change in each transition 5, 10, 15, 20
- Cooling Schedule Additive Linear, Additive Exponential, Additive Trigonometric
- Initial temperature, T_0 250,000, 500,000, 750,000
- Total number of states in the cooling schedule 500, 1000, 2000

By using a sample of 30 random initial policies, S_0, we are able to make a reasonable estimate of performance in practice when it would be expected that many parallel iterations of NSAM would run from random starting points in the decision space. Only a handful, three to five, of values were selected for each of these parameters to get a general sense for their impact on the output. A more thorough sensitivity analysis of the input parameters using principles of rigorous experiment design to determine optimal inputs was beyond the scope of this work due to time limitations; however, it stands to reason that improvements on either run-time and objective function optimality would be evinced with such efforts. The resulting 3,888 design points, each with 30 trials, was completed using 120 CPU cores in parallel in roughly 4 hours.

Table 4.1. Analysis of input parameters into NSAM Vanilla, a basic simulated annealing algorithm, applied to the to the toy problem.

Parameter	Importance
Weibull Shape	0.220
Number of Temperature Steps	0.205
Number of Inter-step Transitions	0.041
Cooling Schedule	0.005
Number of Part Types Changed	0.512
Max number of Infeasible Steps	0.008
Initial Temperature	0.091

Importance values in Table 4.1 were determined by using the output from the input parameters that drive improvements in performance are:

1. The number of part types,i , selected to change when generating each neighbor policy.
2. The shape parameter of the distribution used to select how much to change s_i for each

part type i selected.

3. The number of interstate transitions at each temperature state.

These importance figures were determined by taking the experiment output and building a random forest model ($R^2 = 0.99$) with 10,000 trees, sampling four of the seven variable parameters for each random tree with an observation sampling rate of $n = 10$. With these figures we make the following observations which lead to assumptions for future testing:

1. Cooling schedule selection and the infeasible step limit do not make a significant enough contribution to improving optimality to be considered variable in further testing of NSAM. We set a nominal value for the infeasible steps to 20 and use the additive linear cooling schedule in future testing.
2. Interstate transitions (k), number of part types changed ($count_{S_i}$), and total number of algorithm temperature steps all tend to improve solution quality as they are increased; for future testing, we use a nominal value of 20 inter-step transitions to explore while keeping the number of part type changes[15] to a constant of 15. The values of k and $count_{S_i}$ most significantly contribute to run-times and we maximize their values under given time and resource constraints though in most cases we resort to 2000 transitions and 15 part types respectively.
3. The selected Weibull shape parameter has a significant impact on solution quality. The use of distributions that are skewed left or right based on criteria discussed in Chapter 3 resulted in markedly worse performance than near symmetrical distributions that result from shape parameters above three. For all future testing, shape parameters are set sufficiently high enough to result in near symmetrical distributions.

4.1.2 Results

Given the fast run times of the toy problem, we are able to run a significant number of trials to determine efficacy of NSAM Vanilla in this use case. To determine optimal values for each parameter, a linear regression model was constructed using all main effects and second degree interactions ($R^2 = 0.81$) from the experiment trials. The following parameters are

[15]While not immediately intuitive, selecting 15 parts out of a set of 14 parts does not necessarily mean that all 14 part types will be changed. In our implementation, part types are picked with replacement, therefore it can be shown that with 15 random selections, the expected number of unique part types selected is less than 10.

set for further analysis:

Table 4.2. Values for each of the seven NSAM Vanilla parameters selected for efficacy analysis.

Parameter	Definition	Value
$count_{S_i}$	Number of Part Types Changed	15
λ	Weibull Shape	6.0
n	Number of Temperature Steps	2,000
t_0	Initial Temperature	500,000
k	Number of Inter-step Transitions	20
inf_{max}	Max number of Infeasible Steps	20
$g(t)$	Cooling Schedule	Additive exponential

With these values over 10,000 trials on the toy problem, NSAM results in an improved solution over NAVARM approximately 88% of the time with an average improvement of around 5%. The global optimum is not known and a brute force algorithm would take an impractically long amount of time, therefore there is no way to gauge the overall performance of either algorithm other than to compare their outputs.

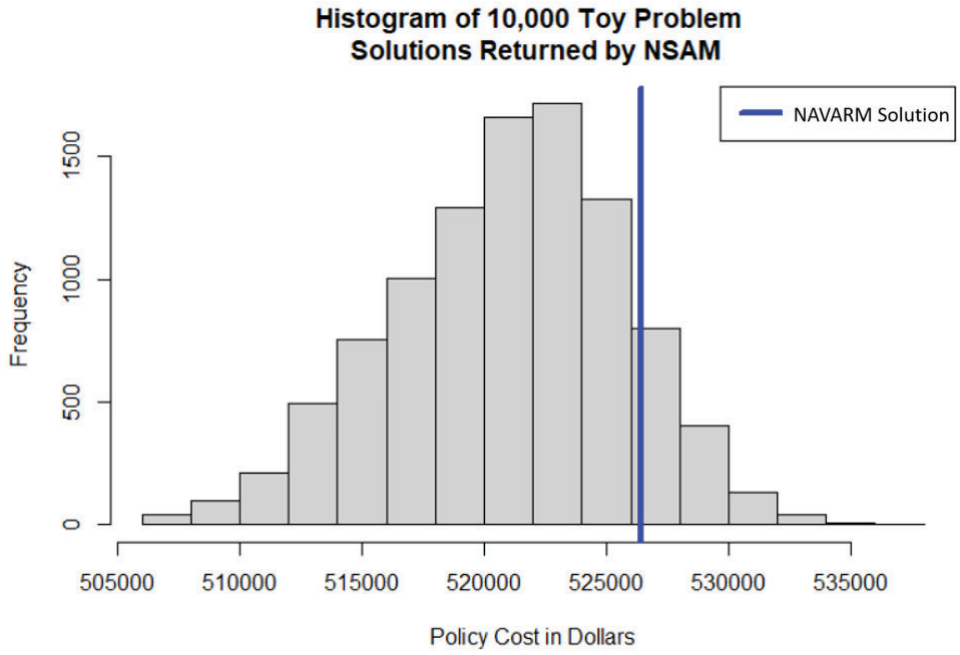

Figure 4.1. Frequency histogram for all results returned by 10,000 trials of NSAM applied to the toy problem. The blue vertical line represents the best feasible value returned via NAVARM's greedy heuristic approach.

Using Figure 4.1 we see that the scale upon which NSAM improves on the NAVARM solution is not large. While NAVARM's solution value lies more than one standard deviation ($\sigma^2 = 4{,}713$) above NSAM's mean output of \$521,013, even the absolute minimum NSAM solution, \$506,714 is only an improvement of about 5%.

4.2 Full Scale CVN RBS Problem

While it may seem out of order, it is most efficient to discuss results of testing NSAM on the full scale RBS problem at this point. The reason for this is that due to the order in which data was made available, our development process actually began with the full scale problem. With this problem, we note interesting behavior with the NSAM algorithm that indicates underlying structure that any MCMC algorithm must contend with when optimizing a multi-indenture problem of this scale[16].

[16]Seven WS and 4,804 part types across 4 levels of indenture

When we first apply NSAM to this full scale problem using our bare bones simulated annealing model, Vanilla, there is a significant optimality gap between various NSAM iterations and results obtained through NAVARM's marginal analysis approach.

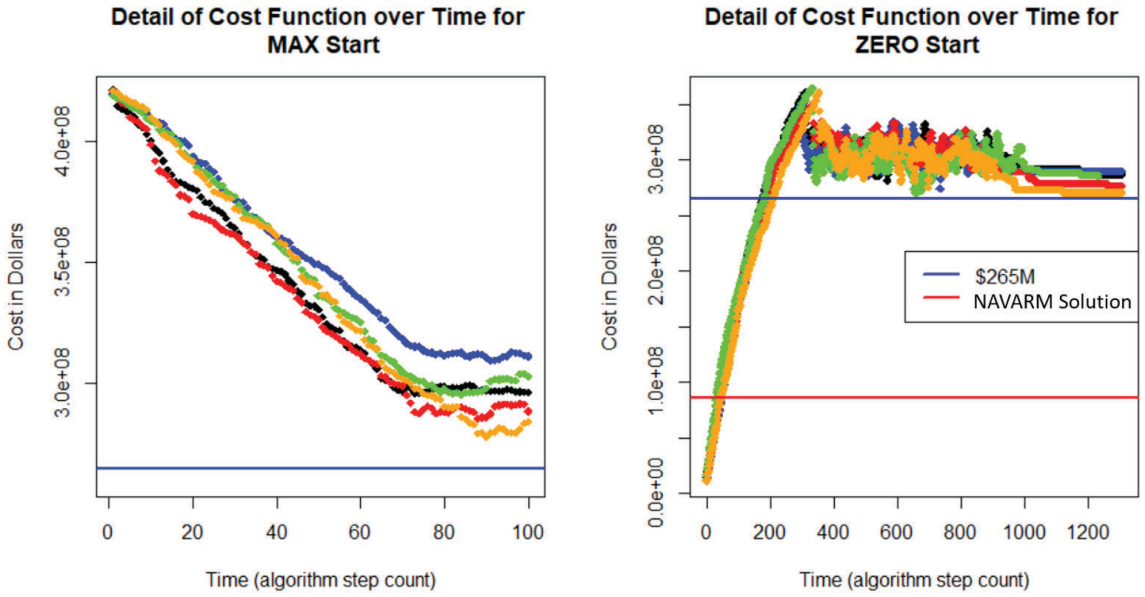

Figure 4.2. The plot on the left is a detail plot of NSAM's first 100 steps for five different trials (represented by the blue, green, orange, black and red points) of the CVN problem when started from all MAX values ($s_i = \bar{s_i} \forall i \in I$). The plot on the left shows the full lifetime of NSAM when started from all zero allowances ($s_i = 0 \forall i \in I$). The blue horizontal line is set at \$265,000,000 on both plots for scale reference. The red horizontal line represents the NAVARM solution to the problem. For both examples the same parameters are used as in Table 4.2 with the exception of $count_{S_i} = 500$ and $t_0 = 50,000,000$.

Figure 4.2 demonstrates the difficulty with the problem structure that exists at this level of complexity. While the Figure 4.2 shows only NSAM starts at ZERO and MAX settings, starting policies selected at random behave in much the same way, what follows is a generalized summary of NSAM's behavior for various starting conditions:

1. Starting policies with cost greater than \$300M: NSAM quickly selects policies that reduce cost until policy cost reaches the vicinity of \$300M. After a period of explo-

ration, the cost functions asymptotically approach a value of approximately $265M.

2. Starting policies that are infeasible: These are typically those with cost less than $265M, however the same behavior is typically exhibited no matter the initial cost if the policy is infeasible. NSAM quickly selects policies with more parts until reaching a feasible solution. This typically happens close to $350M. After a period of exploration, the cost functions asymptotically approach a value of approximately $265M.
3. Though not shown in Figure 4.2, when the initial starting police starts between approximately $265 and $300M, and is feasible, NSAM explores while staying close to the $265M line.

The value $265 has no special significance other than the behavior describes above. What we do know is that there are feasible solutions available at significantly reduced cost, vis-a-vis NAVARM at around $88M. In theory, the problem is ergodic; that is to say that given process by which new policies are selected, all possible states should be accessible by all other possible states. Since ergodicity is extremely difficult to prove on a problem of this scale, we proceed on the assumption that it is.

An exploration of how the two algorithms, NAVARM and NSAM, select parts shows us a possible explanation for this brick wall effect:

Table 4.3. Quantities of SRAs and WRAs part types selected by NSAM and NAVARM.

Part Type	NSAM Count(Value)	NAVARM Count(Value)
WRA	1,206($234M)	1,165($74M)
SRA	1,959($46M)	2,548($14)

What is readily apparent in Table 4.3 is NAVARM is selecting a much wider variety of SRA part types. On average, these are much less expensive than WRA part types. NSAM is selecting a comparable variety of WRA part types however it is selecting WRAs that are much more expensive ($194K average for NSAM versus $63K average for NAVARM) while also selecting more of each part type (1.9 spares on average for NSAM vs. 1.5 for

NAVARM).

This behavior is assumed to occur as a consequence of the multi-indenture structure of the problem. Since EBOs for any given WRA is a sum of the EBO of all its sub-indentured parts, to maintain a constant number of EBO when NSAM selects a reduced quantity of a given WRA, it must also select SRAs sub-indentured to that WRA and then increase their quantity. Vanilla is a bare bones simulated annealing algorithm where part type and quantity selection are random, so the probability of doing this is exceedingly small. For this reason we attempt to apply logic discussed previously for NSAM Chocolate that is meant to overcome this behavior. Due to its large scale, development of Chocolate was not tested using this full scale problem.

4.3 Small RBS Site

For this problem, we are presented with a real-world site with a single WS type. This WS comes with a full scale indenture structure that poses a large challenge for the base simulated annealing model. It was not until testing began at this smaller scale that much of the underlying problem structure was revealed that lead to improvements implemented in Chocolate and Strawberry.

Initial experimentation using the Vanilla simulated annealing algorithm with this problem indicates similar optimality gap to that found the full scale CVN problem. Use of this problem allows for much more rapid testing due to its smaller scale.

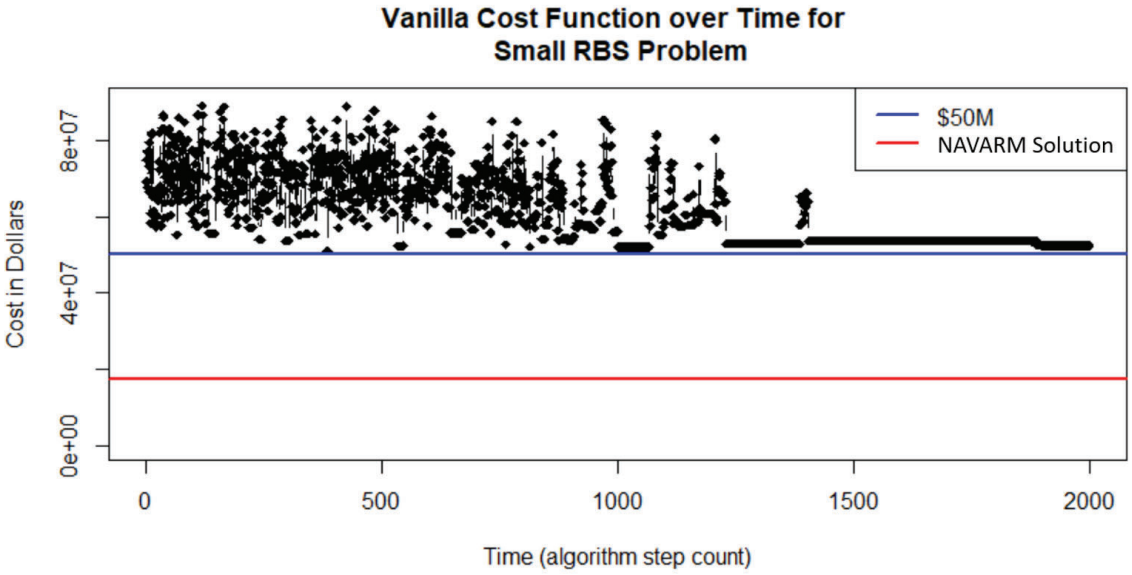

Figure 4.3. Plot of a single trial of NSAM Vanilla applied to the small RBS problem, starting with the first feasible policy. The blue horizontal line is set at $50,000,000. The red horizontal line represents the NAVARM solution to the problem, $17,601,000. Parameters are set as in Table 4.2 with the exception of $count_{S_i} = 100$ and $t_0 = 50,000,00$.

4.3.1 Exploiting the Problem Structure

What we first propose to overcome this WRA/SRA structural difficulty is NSAM Chocolate. This version, as described in Chapter 3, is designed to select sub-indentured SRAs and increase their quantity any time a WRA is selected and its quantity decreased. Other functionality of Chocolate is meant to limit the amount of more complex parts boost the quantity of smaller, cheaper parts at the leaves of the indenture structure.

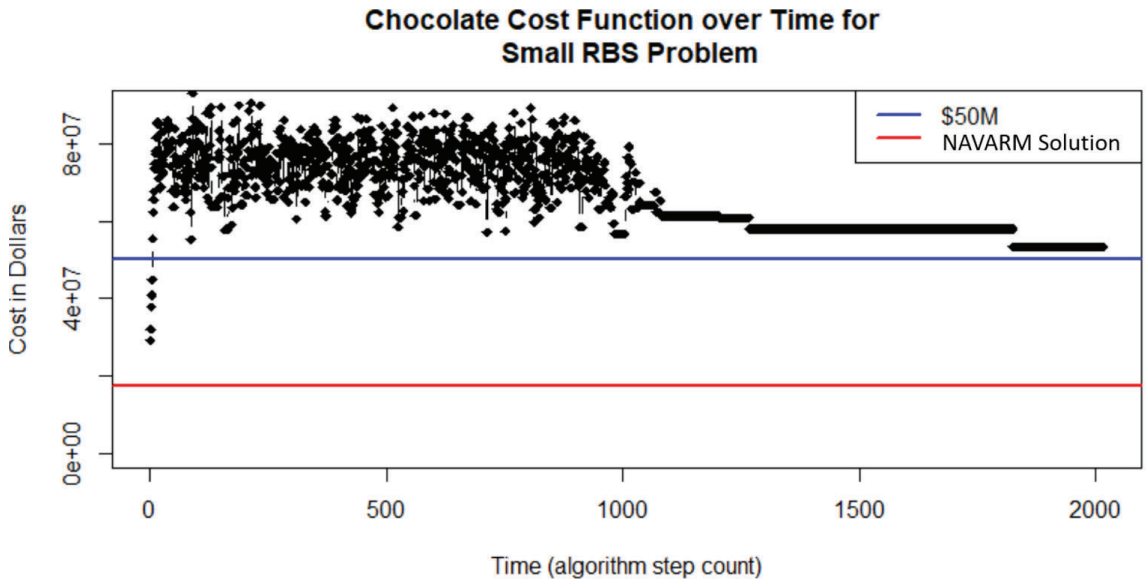

Figure 4.4. Plot of a single trial of NSAM Chocolate applied to the small RBS problem. The blue horizontal line is again set at $50,000,000 and the red horizontal line marks the NAVARM solution to the problem, $17,601,000. Parameters are set as in Table 4.2 with the exception of $count_{S_i} = 100$ and $t_0 = 5000000$.

The outcome of this effort, displayed in Figure 4.4, is no more successful at breaking through the wall as the Vanilla approach. While our attempt to leverage the problem structure to inform the model's selection criteria is ineffective, this does not represent an exhaustive effort and methodologies may exist that are able "break through" the $50M threshold.

Instead of trying to "break through" this threshold, we now focus on a method to begin NSAM with a better solution. We alter Chocolate's logic in the pre-feasible solution building phase to only select parts types without children and WRAs (the distinction is made as not all WRAs have children). This pre-feasible phase is the period before NSAM begins selecting Metropolis moves when the priority is to move from the infeasible region to a policy that is feasible. A feasible solution must be found before NSAM can start as the acceptance criteria requires a feasible incumbent solution with which to compare selected neighbor policies.

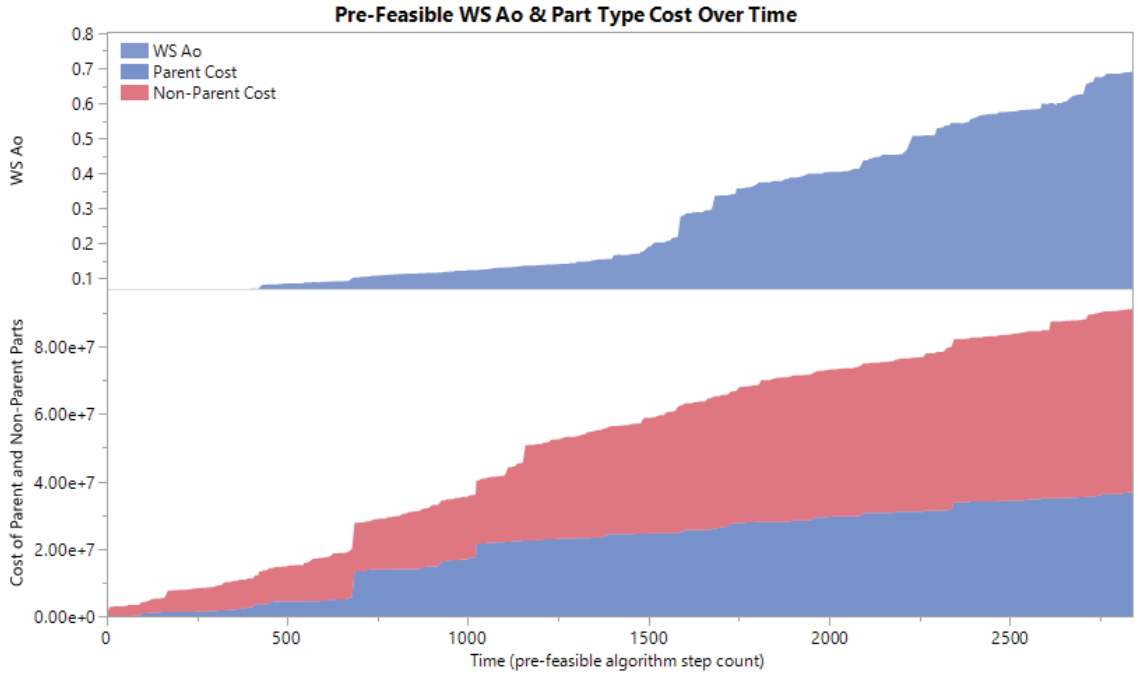

Figure 4.5. The top plot indicates WS A_o as the initial policy is constructed using Chocolate's logic. The bottom stacked plot shows the individual and total costs of parent and non-parent parts selected over the same period.

In Figure 4.5 we see steps where there is a significant jump in cost without an appreciable increase in A_o. These points, such as at step 682, demonstrate very inefficient choices by the algorithm that even the aggressive selection logic is unable to prevent. What is needed is a method by which an initial stocking policy is constructed using the most efficient method possible. This leads us to the final iteration of NSAM, Strawberry.

4.3.2 Epsilon Greedy Approach

The most efficient method currently available is the same method used by NAVARM to construct its output policies. Marginal analysis, or NAVARM's greedy heuristic, discussed at length in Chapter 2, selects parts from a sorted list of greedy ratios. Our efforts so far have validated the efficacy of this approach by the existence of the large optimality gaps that separate NSAM solutions from NAVARM's.

The epsilon greedy approach, discussed in the NSAM Strawberry section in Chapter 2,

induces a small ($1 - \epsilon$ where typical values of ϵ are greater than 0.9) randomness in the selection process and recalculating greedy ratios to approach the initial feasible solution from a slightly different position in the state space.

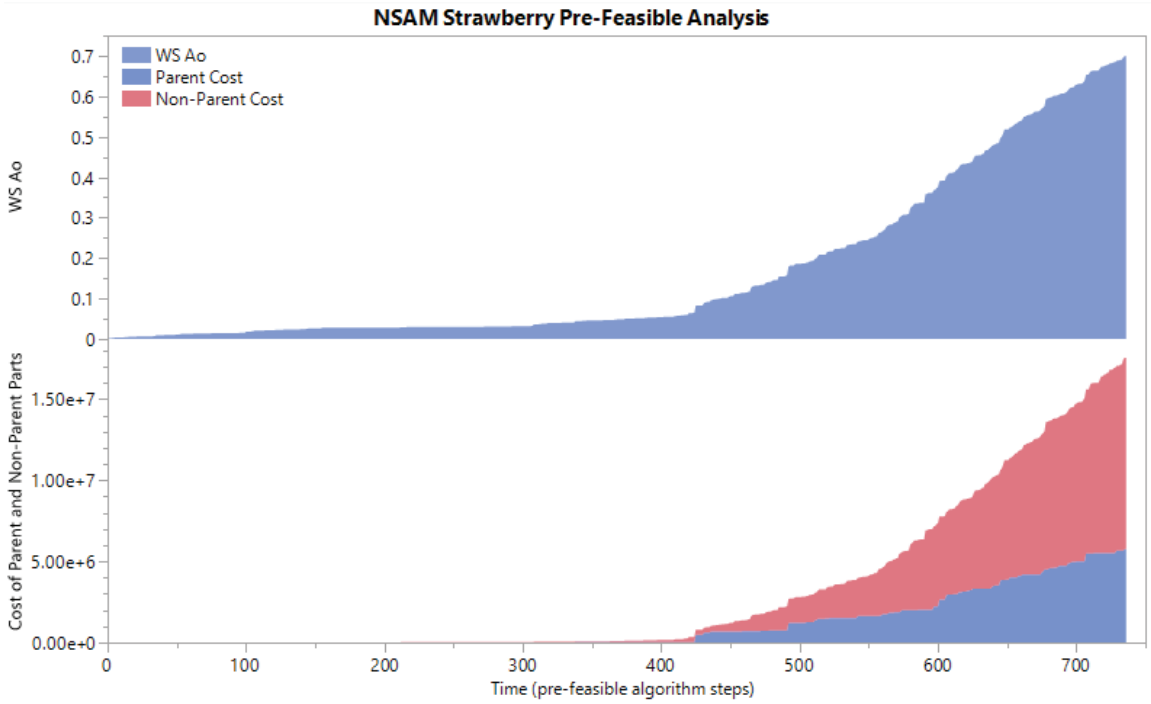

Figure 4.6. The top plot indicates WS A_o as the initial policy is constructed using Strawberry's epsilon greedy approach. The bottom stacked plot shows the individual and total costs of parent and non-parent parts selected over the same period. For this trial, value of 0.001 was used for epsilon.

Comparing Figure 4.5 with Figure 4.6 there is a notable difference in how the policy is constructed using the epsilon greedy approach. The trial that resulted in the policy illustrated in Figure 4.6 used an epsilon value of 0.001. This particular run resulted in an initial feasible solution value of $17,619,672.79, which is not an improvement over the NAVARM value of $17,601,339.62. However, once the initial feasible solution was reached, the simple NSAM Vanilla logic improved the solution to a value of $17,548,044.87 which improves on the NAVARM solution.

Table 4.4. Pre and Post Feasible Solution Optimality by ϵ value.

ϵ	Quantity of Pre-Feasible Trials < NAVARM SOL	Quantity of Post-Feasible Trials < NAVARM SOL	Total Trials
0.999	93	94	176
0.995	9	13	187
0.99	1	1	140
Σ	103	108	503

A total of 503 trials were conducted using three different values for epsilon. By far, the smaller values of epsilon performed better. At 0.001, the expected number of non-greedy actions taken in construction of the policy is less than 1 (since the policy is constructed in under 1000 moves); however, even with that being the case, as evidenced in Table 4.4, the initial solution produced with this value of epsilon improved on the NAVARM solution over 50% of the time though with typical improvements not exceeding 0.01% of total policy cost but with multiple instances of policies that save tens of thousands of dollars.

Once constructed, however, across all values of epsilon, NSAM only improved 26 solutions with all other solutions getting worse (meaning the initial solution was the best value found in those runs). This does well to illustrate the highly complex nature of the state space in the problem.

4.4 CVN Problem Revisited

While run times for NSAM at this scale are reasonable, about a minute per trial from start to finish, run times are significantly higher at larger scale. This is due to the fact that many more parts must be selected each step, as well as a significantly increased processing time required to calculate greedy ratios and EBOs for a much larger problem. For that reason only 70 trials are run on the CVN problem using NSAM Strawberry:

Table 4.5. Pre and Post Feasible Solution Optimality by ϵ value

ϵ	Quantity of Pre-Feasible Trials < NAVARM SOL	Quantity of Post-Feasible Trials < NAVARM SOL	Total Trials
0.999	0	0	35
0.9998	0	0	35
Σ	0	0	70

While none of the trials managed to find a better solution than strict marginal analysis via NAVARM, the solutions found were all between $100M and $125M (NAVARM solution for the CVN problem is approximately $85M). This is a significant improvement over the Vanilla approach which had typical optima not better than $265M. While this attempt did not manage to improve on the incumbent algorithm, there is much potential for improvement with so many parameters that need further analysis to provide optimal performance.

CHAPTER 5:
Summary and Future Work

Having thoroughly discussed our methodology and the underlying principles used therein, we conclude with a summary of our findings. We begin with how our results validate the efficacy and design of the incumbent algorithm used to optimize the RBS problem. We then discuss the utility of simulated annealing in the same context and how our results can be leveraged to improve solutions currently obtained by marginal analysis or with an epsilon-greedy approach to marginal analysis. We conclude with our recommendations for future work and some final thoughts on our research.

5.1 Incumbent Validation

The first conclusion we draw from this work is a validation of the methodology underpinning NAVARM. Given the performance of basic simulated annealing and in particular the large optimality gap that emerges when the problem scales from our toy problem to the smallest possible real world scenario, it is apparent that the best way to deal with assembling inexpensive, near-optimal solutions relies on cost efficient selection criteria. Marginal analysis is predicated on making efficient selections, one step at a time, making it well suited for the purpose of "evading" the wall of complexity that is responsible for the gap in the first place.

One note on NAVARM is that, as written, it only calculates greedy ratios once per iteration. This is a deviation from the methodology described by Sherbrooke 2004 who describes a method by which the EBOs are recalculated as parts are added to the policy. In experimentation with NSAM Strawberry, we find that frequent recalculation of EBO is inexpensive at smaller scale, but at large scales, such as the CVN problem, this recalculation done thousands of times leads to very long run times when executed using NSAM software[17].

[17]NSAM Strawberry creates NAVARM solutions when $\epsilon = 0$ since the greedy action, being performed 100% of the time, is executing NAVARM's sorted greedy ratio list

5.2 Simulated Annealing as an Optimization Method

Our success with NSAM's base simulated annealing algorithm at the small and medium scales provide evidence that under the right conditions, an MCMC method is capable of outperforming marginal analysis, which is the current gold standard for multi-indenture optimization. The problem then becomes setting the right conditions for the algorithm to flourish. First, to define the conditions we must identify the bounds of the optimality gap between the region of highest known quality (presumed to be in the vicinity of NAVARM solutions) and the region the MCMC algorithm tends towards. In our CVN case, these are the regions defined by policies where the cost function is in the vicinity of $85M and $265M, respectively.

In order to be exploited, a method to initialize solutions in a promising region must be devised. One such method, already used in practice by NAVARM is marginal analysis. By inducing a small amount of randomness we have demonstrated that it is possible to initialize solutions that are better than those found by marginal analysis on its own, however we have also shown that it is possible to improve solutions initialized in this region to be better using simulated annealing as well.

So while it may be possible to construct logic to eventually find a path from one region to the other, as attempted unsuccessfully with NSAM Chocolate, the most likely application of simulated annealing is as a way to polish solutions generated using other algorithms. Using parallel processing, many separate chains can be initialized from the same point or separate points generated randomly, as with our epsilon greedy approach. We have demonstrated that improved solutions can be found in a very reasonable time-frame on the small RBS problem, however due time constraints on our research and limitations of our software we have not yet applied simulated annealing to a large scale problem such as the CVN problem.

We have also demonstrated generalizability of the simulated annealing algorithm in this setting. The iterative approach to our design used different approaches to calculating A_o and making numerous other design choices that were changed in the process of scaling the algorithm. However, no matter what changes were made, the cooling schedule - move class - acceptance loop remained at the center of the optimization algorithm. Even if the practitioner were to completely change the constraints or distributional assumptions, it could be done simply and without impacting the primary mechanics of the algorithm. This makes

a strong case for simulated annealing as a generalizable tool for inventory optimization. For example, it is conceivable that an organization may desire to measure readiness using some other metric that could only be obtained using simulation. The output of the simulations could be easily applied the constraints on the simulated annealing algorithm at each step as feasibility is a separate module that merely informs the basic class – cooling – acceptance loop.

5.3 Epsilon-Greedy Approach to Policy Initialization

The other significant finding resulting from this research is that introducing a small amount of randomness into the marginal analysis algorithm, we are able to achieve better policies than with marginal analysis alone. Since this approach does not guarantee improved solutions, practical application requires the marginal analysis solution, using NAVARM or $\epsilon = 0$, and many runs of an epsilon-greedy policy building algorithm such as NSAM Strawberry. We have not yet confirmed this for the large CVN type problem due to time constraints, so this should also be examined when considering future research.

5.4 Future Research

Input Parameter Optimization

In our experimentation, we use rather rudimentary methodology to determine optimal input parameters for NSAM. It is expected that every problem will have variance in what parameters yield the most optimal performance from the algorithm; however, an underlying commonality exists between all problems in our research. Examples of this commonality are:

- As the number of states increases, so does the optimality of the final output policies.
- The ideal number of part types changed each iteration is large in proportion to the total number of different part types.
- The best performing initial starting temperatures are those that allow for expected acceptance probabilities of around 0.9.

These are simply patterns noted in experimentation, however a more exhaustive approach

to optimizing these inputs for various problems is merited (and, if the results can be generalized, all the better).

Move Class Improvement

The rudimentary Metropolis move used in NSAM is an apparent shortcoming of the algorithm. Presumably, with the right move class design, the existence of the "brick wall" or optimality gap we discover in the larger scale problems would not impede the progression of policy selection. Other MCMC methods offer alternative approaches to the move class that may be able to navigate the "brick wall." Another possibility is to refine the NSAM chocolate approach to further limit the selection criteria or otherwise inform the algorithm to guide it toward desired behavior.

In addition, linking many of the algorithm criteria to the temperature is another area of possible improvement. Currently NSAM sets all parameters constant for the entirety of the process. What is proposed is to change some of these as a function of temperature. For instance, the number of part types changed each step starts large but as the system cools, fewer part types are changed. This would allow for increased exploration in the early stages and increased exploitation in the latter.

Apply Simulated Annealing to Selected Large Scale Initial Solutions

While validated in the toy and small RBS site problems, an unfortunate lack of time prevents the scope of this research to test simulated annealing on the large scale CVN problem from desired initial solutions. An analysis of performance at this scale is necessary to definitively state that polishing already near-optimal solutions will yield improved solutions.

Limiting Probability A_o Calculation

Given that the most expensive operation in the NSAM algorithm is the calculation of EBOs (and consequently A_o for each WS) that occurs at every step, it follows that optimizing this calculation has obvious benefit. In Chapter 3 we propose an alternative method of calculating A_o using Markovian birth-death processes where the limiting probabilities are calculated and used to estimate the proportion of time that aircraft are operational. While our use of this method is transitory and only used at the smallest scales in our development process, there

is potential for this method to be useful in performance optimization. Given that the method relies on many large matrix operations, use of graphics processing unit architecture could provide large performance gains as measured by reduced resource requirements. An analysis of output quality compared to the current A_o calculation methodology is also necessary to validate the methodology.

5.5 Conclusion

Sherbrooke's marginal analysis approach for multi-indenture models was first utilized when computing power was not ideal for an MCMC type approach for problems of this scale. Therefore what was first envisioned for this project was to find an inventory optimization method by which we could improve on the current practice by leveraging the availability of today's increased computational resources. We instead succeeded in validating Sherbrooke's approach as implemented by Salmeron and Buss in NAVARM. Given the promising application of simulated annealing as a polishing algorithm for near-optimal solutions, and the potential for improved solutions using an epsilon-greedy algorithm, we are confident that this field will remain fertile ground for research for the foreseeable future.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Burdick L (1991) Readiness based sparing (RBS): From concept exploration to full scale development. *Naval Engineers Journal* 103(2):71–82.

Bynum ML, Hackebeil GA, Hart WE, Laird CD, Nicholson BL, Siirola JD, Watson JP, Woodruff DL (2021) *Pyomo–Optimization Modeling in python*, volume 67 (Springer Science & Business Media), third edition.

Congressional Budget Office (2022) Availability and use of aircraft in the air force and navy. https://www.cbo.gov/system/files/2022-01/57433-aircraft.pdf.

Department of the Navy (2021) Department of the Navy fiscal year (FY) 2022 Budget Estimates, Justification of Estimates may 2021. https://www.secnav.navy.mil/fmc/fmb/Documents/22pres/OMN_Book.pdf.

Eckstein M (2020) Mission capable: How the navy harnessed its data to achieve 80 percent fighter readiness. United States Navy, Office of the Chief of Naval Operations, URL https://p2p.navy.mil/Media/News/Article/2164733/mission-capable-how-the-navy-harnessed-its-data-to-achieve-80-fighter-readiness/.

Graves SC (1985) A multi-echelon inventory model for a repairable item with one-for-one replenishment. *Management science* 31(10):1247–1256.

Gurobi Optimization, LLC (2022) Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.

Hart WE, Watson JP, Woodruff DL (2011) Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation* 3(3):219–260.

Hastings W (1970) Monte carlo sampling methods using markov chains and their applications. *Biometrika* 57(1):97–109.

Katz J (2022) Navy air boss has new aircraft readiness targets to hit. Breaking Defense, URL https://breakingdefense.com/2022/02/navy-air-boss-has-new-aircraft-readiness-rate-north-stars-to-follow/.

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.

Mehta A (2018) Mattis orders fighter jet readiness to jump to 80 percent — in one year. URL https://www.defensenews.com/air/2018/10/09/mattis-orders-fighter-jet-readiness-to-jump-to-80-percent-in-one-year/.

Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6):671–680.

Oracle (2014) Java™ platform, standard edition 8.

Rardin RL (2019) *Optimization in Operations Research* (Pearson), second edition.

Ross SM (1997) *Introduction to Probability Models* (San Diego, CA, USA: Academic Press), sixth edition.

Salmeron J, Buss A (2021) Project deliverable: Naval aviation readiness-based sparing model – release 2.6. Naval Postgraduate School, Monterey, CA, provided by J. Salmeron with approval from NAVSUP-WSS Philadelphia.

Saloman P, Sibani P, Frost R (2002) *Facts, Conjectures, and Improvements for Simulated Annealing* (Society for Industrial and Applied Mathematics).

Sherbrooke C (2004) *Optimal Inventory Modeling of Systems, Multi-Echelon Techniques, Second Edition* (Kluwer Academic Publishers, Boston).

Sherbrooke CC (1966) METRIC: A Multi-Echelon Technique for Recoverable Item Control. Technical report, RAND Corp, Santa Monica CA.

Sherbrooke CC (1971) An evaluator for the number of operationally ready aircraft in a multilevel supply system. *Operations research* 19(3):618–635.

Slay M (1980) *VARI-METRIC: An Approach to Modeling Multi-echelon Resupply when the Demand Process is Poisson with Gamma Prior. Working Paper.*

White SR (1984) Concepts of scale in simulated annealing. *AIP conference proceedings* (122):261–270.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California