

ANALYZING SPARING POLICY IN THE OPERATIONS OF SPACE HABITATS

A Dissertation
Presented to
The Academic Faculty

by

Andrew Jones Maxwell

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2023

COPYRIGHT © 2023 BY ANDREW JONES MAXWELL

ANALYZING SPARING POLICY IN THE OPERATIONS OF SPACE HABITATS

Approved by:

Dr. Koki Ho, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Brian Gunter
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Alan Wilhite
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Robert Moses
Langley Research Center
*National Aeronautics and Space
Administration (retired)*

Dr. Glenn Lightsey
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: [April 18, 2023]

ACKNOWLEDGEMENTS

To my wife, Adair, thank you for everything. Without your patience over this too long process, it would have ended before finishing. It is weird to think that I've been working on this as long as I've known you, but I can't wait to see what the future has in store for us.

I would like to thank my mother for her unceasing belief and support. And to my father, thank you for the counsel and insight through this whole process.

Thank you to Dr. Wilhite for starting me on this path and for giving me the opportunity to study things that few get to consider. I also owe a debt of gratitude to Dr. Ho for the guidance he gave to get to this point.

I would also like to thank Kandyce Goodliff, Bill Cirillo, and Chel Stromgren for introducing me to the EMAT tool and helping to initiate this research. And I'll never be able to thank Dennis Bushnell enough for expanding my horizons of what the past and future of spaceflight have to offer.

Thank you to Scott Gustafson for providing a helping hand at a critical juncture. And finally, thank you Tim, Patrick, Chris, and the rest of my NIA compatriots for your help along the way.

TABLE OF CONTENTS

| | |
|---|-------------|
| ACKNOWLEDGEMENTS | iii |
| LIST OF TABLES | vi |
| LIST OF FIGURES | vii |
| LIST OF SYMBOLS AND ABBREVIATIONS | viii |
| SUMMARY | xii |
| CHAPTER 1. Introduction | 1 |
| 1.1 Previous Approaches | 4 |
| 1.1.1 Exploration Maintainability Analysis Tool | 4 |
| 1.1.2 Semi-Markov Process | 5 |
| 1.1.3 Knapsack Problem | 7 |
| 1.1.4 Including Commonality | 8 |
| 1.1.5 Sparing Policies | 9 |
| 1.2 Research Objective | 10 |
| CHAPTER 2. Motivating Case Study | 12 |
| CHAPTER 3. A simulation approach for testing sparing Policies | 19 |
| 3.1 Simulation Methods | 19 |
| 3.1.1 Existing EMAT | 20 |
| 3.1.2 Optimal Spares Allocation | 25 |
| 3.1.3 Implementing Testing of Sparing Policies with a Failure Queue | 31 |
| 3.2 Comparison of Optimization Approaches | 35 |
| 3.3 Comparison of Sparing Approaches | 37 |
| 3.3.1 Comparison Between the Baseline Policy and Lazy Policy | 37 |
| 3.3.2 Impact of Sparing Policy Changes on Alternative Configurations | 41 |
| 3.4 Conclusion | 49 |
| CHAPTER 4. An Analytical Model for Sparing Policy Analysis and Optimization for Space Habitat Operations | 50 |
| 4.1 Analytical Methods | 50 |
| 4.1.1 Structuring the Semi-Markov Process for Sparing Policy Analysis | 54 |
| 4.2 Optimizing Spare Policy | 68 |
| 4.2.1 Initializing the Population | 70 |
| 4.2.2 Particle Swarm Optimization | 72 |
| 4.3 Results | 74 |
| 4.4 Conclusion | 79 |
| CHAPTER 5. Conclusion | 80 |
| 5.1 Possible Improvements | 84 |
| 5.2 Areas of Future of Investigation | 88 |

| | | |
|-------------------|----------------------------------|-----------|
| 5.2.1 | Additive Manufacturing | 88 |
| 5.2.2 | Cannibalization | 90 |
| 5.2.3 | Preventative Maintenance | 91 |
| 5.2.4 | Super Heavy Lift Launch Vehicles | 92 |
| REFERENCES | | 94 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Component Parameters | 15 |
| Table 2: Optimization Approach Comparison Parameters | 35 |
| Table 3: Sparing Policy Comparison Parameters | 38 |
| Table 4: Comparison of spares policies' allocation at 99.9% demand satisfaction | 40 |
| Table 5: Transition Rules | 55 |
| Table 6: Probabilities of Sufficiency for Different Sparing Policies | 77 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1: Carbon Dioxide Removal Assembly (CDRA) | 13 |
| Figure 2: Lithium Hydroxide (LiOH) canister | 14 |
| Figure 3: Time step flow of the EMAT tool | 21 |
| Figure 4: Time step flow modified for generic sparing policies | 33 |
| Figure 5: Comparison between the modified knapsack approach and the Greedy algorithm | 36 |
| Figure 6: Comparison between the baseline and lazy policy | 39 |
| Figure 7: Alternative days to repair results compared | 43 |
| Figure 8: Lazy policy with 5 times days to repair | 45 |
| Figure 9: Additional strings results compared | 46 |
| Figure 10: Lazy policy with component failure rate doubled | 48 |
| Figure 11: General analysis flow | 53 |
| Figure 12: System Configuration | 60 |
| Figure 13: State Network Diagram after 11 | 61 |
| Figure 14: State Network Diagram after -11 | 62 |
| Figure 15: State Network Diagram after 1-1 | 63 |
| Figure 16: Comparison of analytical and simulation results | 75 |

LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|-------------------------------------|--|
| <i>allocStorage</i> | spares allocation storage array |
| <i>c</i> | maximum capacity of spacecraft, kg |
| <i>c₁, c₂</i> | particle swarm confidence constants |
| <i>CDF</i> | Cumulative Distribution Function |
| <i>convper</i> | convergence percentage change criterion |
| <i>D</i> | total mass steps |
| <i>f</i> | large integer constant |
| <i>f</i> | difference distribution matrix/vectors |
| <i>F</i> | difference CDF matrix/vectors |
| <i>g</i> | PDF of first passage time matrix |
| <i>H</i> | unconditional waiting time density matrix |
| <i>i</i> | iteration block counter |
| <i>I</i> | Identity Matrix |
| <i>ite</i> | iteration count |
| <i>iterblock</i> | block of iterations between convergence checks |
| <i>k</i> | number of visits to a state |
| <i>l</i> | total number of possible spares |
| <i>m</i> | mass of each spare type, kg |
| <i>M</i> | total spares mass, kg |
| <i>massdis</i> | mass discretization, kg |
| μ | mean of lognormal distribution |
| <i>n</i> | number of elements in the system |

| | |
|--------------------------|--|
| N | number of spare types |
| p | position of previously found optimum |
| PDF | Probability Distribution Function |
| PoS | probability of sufficiency |
| $posTable$ | storage array for POS |
| Q | Kernel matrix |
| res | mission simulation result allocation |
| σ | standard deviation of lognormal distribution |
| s | Laplace input variable |
| $spareinv$ | simulation spare inventory |
| $S(x)$ | CDF of spares demanded |
| $SMiss$ | number of successful missions |
| $SpareFailCount$ | number of times each spare contributed to a failed mission |
| $SpareImprovementMetric$ | spare improvement metric |
| t | time (s) |
| T | mission duration, day |
| $TotalSims$ | total number of simulations |
| V | Markov Renewal Process Probability |
| v | particle velocity |
| w | particle swarm inertia factor |
| x | quantity of a given spare (in MKP context) |
| x | individual particle position (in PSO context) |
| X | set of spares |
| ζ | repair or buffer depletion transition parameter |

\vec{Z} set of transition parameters

Subscripts and
Superscripts

A analytical

b number of new spare type being added to allocation

bd buffer depletion

c current

d mass step

$final$ final

g global

i source state

$initial$ initial

j destination state

k spare type (if using 3rd dimension)

k particle swarm iteration

l particle number

MKP, A Modified Knapsack Problem, Analytical

MKP, S Modified Knapsack Problem, Simulation

n transition id number

$no\ spares$ no spares utilized

opt optimal

p parameter index

s spare type

sim simulation

t target

T mission duration

0 iteration 0 (initial)

\sim function or variable is in the Laplace domain

SUMMARY

The inclusion of operational sparing policies in early system definition can ensure that spares allocations can optimally meet desired system reliabilities consistent with the planned maintenance of a crewed vehicle. This approach is critical for long-duration crewed missions where mass allocations are constrained and lack of safe abort contingencies limit options in the event of significant system degradation, especially in the environmental and life support systems. The research presents both simulation-based and analytical approaches to identify optimal spares allocations for various sparing policies.

The developed simulation method improves a current state-of-the-art tool from two perspectives. First, it develops a new method based on the modified knapsack problem to generate the spares allocations that maximize the probability of sufficiency given a mass capacity. Additionally, it develops a simulation model with a failure queue to enhance the flexibility of the state-of-the-art model to evaluate different sparing policies. With the developed method, comparative studies using two allocation approaches and two different policies are presented.

The research also presents an analytical model for analyzing and optimizing sparing policies as part of an overall evaluation of the probability of sufficiency for a system configuration. The repair transition parameters are varied to change the state visitation probabilities which drive a change in the probability of sufficiency observed for a given mass allocation. These parameters are optimized using a particle swarm optimizer to identify the preferred strategy for a desired allocation mass.

CHAPTER 1. INTRODUCTION

Crewed missions to Mars represent a new challenge when it comes to system reliability and the spares necessary to achieve mission success. The orbital mechanics of the mission and the limits of near-term propulsion technology meaning that multi-year mission durations are unavoidable[1]. Additionally, the geometries of Earth, Mars, and the trajectories between them during the mission foreclose on quick abort options like those available for the International Space Station (ISS) and the Apollo missions [2]. In other words, long duration human spaceflight, especially that which extends beyond cislunar space, will require a balance of highly reliable systems, adequate spares, and intelligent logistics planning to ensure successful outcomes for the crew and mission. It is estimated that for every additional kilogram that is delivered to the surface of Mars an additional 100kg must be delivered to low Earth Orbit [2] meaning that decisions regarding the amount of spares to allocate can have a significant impact on the overall architecture design. This estimate is consistent with the finding that the Apollo program required 165 kg on the launch pad for each kg on the command module [3] resulting in an extreme mass reduction program of scraping the excess material of structural elements to save grams of mass [4]. Given that a high probability of crew survival is an anticipated requirement for these types of systems [1], the reliability should be included as part of any architectural trade and should be tied to the spare allocations used as part of the candidate systems within the development of design reference missions.

Past reference architectures for human Mars missions [1,5] identified the need for highly reliable systems but proposed reference solutions whose sparing allocations are not

traceable to a system-level measure of success. For example, DRA5.0 [1] provisions roughly 20% of non-structure mass of the in-space transit habitat for spares without identifying the implied reliability of the allocation. As a result, the solution presented, and ones like it, have a latent risk that the system architecture does not actually meet the desired requirements. The selection of a baseline solution that does not meet requirements at the outset may result in late design changes, forced acceptance of higher risk, or deferment of technology development investment due to a misjudgment of capability. Thus, we need to estimate the optimal spares mass for the baseline system configuration to ensure that the system can meet the desired requirements.

These concerns are not merely theoretical. During the development of the Space Transportation System, commonly known as the Space Shuttle, subjective estimates for catastrophic failure were between 1 in 100 and 1 in 100,000 [6]. Following actual catastrophic failure of the Shuttle, a quantitative probabilistic risk assessment was created, and the failure estimates were updated to between 1 in 10 and 1 in 90, depending on the actual flight configuration used [7]. These findings could not inform the development of the system because they were not generated until long after the Shuttle entered operations and had experienced loss of crew. Evaluating expected the performance of the system from a reliability perspective should occur as early in the design as possible so that decision makers can make informed decisions about both the design of the system and the planned spares allocation.

In performing such an assessment, different sparing policies should be considered when determining an optimal spares mass for a given system. Traditional methods typically use a repair-on-failure sparing methodology that may be overly conservative especially for

systems with internal redundancy, which is typical for environmental control and life support (ECLS) systems utilized in crewed habitats. Additionally, it is unclear what sparing approach will be used in practice since there is no direct analog for a multi-year, fixed duration human spaceflight mission with no abort options. Explicitly including the sparing policy in the analysis not only serves to clearly set the baseline but also gives visibility to an operational mitigation that can be considered up until the end of the mission. By exposing more of the degrees of freedom during the conceptual design phase, system performance can be better assessed against the desired capabilities and the state of the design can be better understood.

For the assessment to be meaningful, the measure of performance must be adequately addressed and capture the impact of the inputs under consideration. For questions of reliability and the spares needed to achieve a desired one, this means exploring the Pareto frontier between the probability of sufficiency (PoS) and the spares mass. Probability of sufficiency was chosen because it measures the probability that the allocated spares are adequate to meet the demands of spares driven by failures in the system and the associated sparing policy to allow the mission to complete. This metric is a broader measure than reliability which traditionally measures the ability of an item to function over a given period of time without repair. Using probability of sufficiency helps to capture the reliability of the system combined with repair actions with available spares. With the metric selected, the developed method needs to achieve a balance between configuration realism, computational quickness for tradespace exploration, and acknowledgment that the final configuration is not known.

1.1 Previous Approaches

Several methods have been employed to estimate the reliability and spares needs of space systems. Some analyses relied on the assumption that the system is comprised of elements at such a high-level of repair that they can be treated as serial configurations [8,9] which is not compatible with problem under consideration. However, a few approaches were identified as potential candidates to serve as the basis for an approach for identifying both the PoS of a system and for testing different repair policies against the same configuration.

1.1.1 Exploration Maintainability Analysis Tool

One of the relevant state-of-the-art tools used by the National Aeronautics and Space Administration (NASA) for maintainability analysis is the Exploration Maintainability Analysis Tool (EMAT). EMAT is a discrete event simulation-based tool for determining the expected probability of sufficiency for a spare allocation. This tool leverages existing fault tree analysis to model failure, repair, and consumable use on a time step by time step basis. This approach is used with Monte Carlo to generate many simulation outcomes. Typically, the tool is initialized with some consumable spares and the results of a given Monte Carlo run were used to determine the next optimal spare to add [10]. This process for adding spares is then repeated until the allocation tested has a high probability of being sufficient to support the mission is obtained. This greedy strategy assumes that in all cases the previous spares allocation is a subset of the next one whether or not that is strictly true.

As a discrete event simulation-based tool, it is possible to update the underlying events and how they are triggered to test sparing policies. Previously [10], the only sparing approach tested was a repair-on-failure policy which initiates a repair action immediately upon failure of a component. This behavior allows comparison to series-only estimation methods but has the potential to be overly conservative, especially if the larger system has internal redundancy, which can support continual function of the system in the event of certain component failures instead of repair. The simulation can be updated to employ a different rule set to initiate and complete repairs, but those rules must be formulated explicitly and implemented in the underlying code of the simulation. An example of this will be done as part of the simulation methodology in this work.

1.1.2 Semi-Markov Process

While EMAT uses a simulation to generate possible outcomes, Owens [11] proposed an approach using a Semi-Markov Process (SMP) to model states of the system and estimate the time spent in each state. Any system can be modeled as an SMP if it can be described using a set of states with random variables governing the occurrence of transitions between the states. This formulation is especially useful when there are multiple options to transition from any given state of the system and there is a desire to know how often one transition occurs versus another available one. Unlike Markov chains, the transitions between states can be modeled using any type of probability distribution or mass function. Pyke [12,13] proposed general solutions to SMP-based problems utilizing the Laplace domain. Warr and Collins [14] described a solution for finite-state SMPs using numeric inversion of Laplace transforms which Owens [11] extended to space systems. Additionally, Owens identified that non-memoryless transitions between states would result in difference distributions that

could also be solved with numeric convolution. This approach is compatible with the method used to perform the Laplace transforms, which also requires discretizing the continuous random variables describing basic event transitions. Transitions could not be assumed to be memoryless because repair actions and consumable buffer depletions are not suited to be modeled with a constant rate or other transition whose likelihood of occurrence is independent of how much time has elapsed. The numerical inversion of the Laplace combined with convolution for the difference distributions formed the basis of the analytical approach used to explore sparing policies.

Like most Markovian approaches, SMP is challenged by state space explosion. Owens attempts to mitigate this by limiting the total number of failures that could accumulate before repair returned the system to the initial state, called renewal for a SMP. Even with this valid simplifying assumption, the six element model results in 91 states. The analytical approach in this work adopts this approach to mitigate state space explosion and leverages advances in modern computing to work with state network diagrams containing over 1,500 states. To enumerate such a large state space, an auto-generation method is implemented. Previously, Owens and de Weck [15] describe a general algorithm for doing this but do not detail how this is routinized for computer processing. Since the case study of interest was such a large problem, an independent method for automatically generating was developed and implemented. This approach is described in Chapter 2.

SMP is utilized in this work as the basis of an analytical approach to rapidly identify promising sparing policies. It does this for a given configuration by turning the sparing policy from a set of mechanistic rules underlying a simulation into independent variables. Specifically, the parameters that determine the shape of the repair and buffer depletion

random variables become the inputs to an optimization problem. By searching for the set of random variable parameters that maximize PoS for a given spares allocation mass, better spare policies can be found for the specific system under configuration rather than relying on the imagination of the modeler to come up with the best set of rules.

1.1.3 Knapsack Problem

Classical methods for designing an optimal spares allocation for long-duration missions are reviewed in Messinger and Shooman [16], which proposed a dynamic programming method for identifying optimal spares allocations. This approach directly leverages solutions to solve the knapsack problem, a classic formulation of and solution to the discrete optimization problem of how to add items to a backpack to maximize the system value subject to the resource constraints. Selecting the combination of spares that add the most reliability to a system is a direct analogy. To determine the reliability of a given combination, Messinger and Shooman utilized a reliability model that grouped similar components and treated unlike components as independent and in a serial configuration. This structure along with a constant hazard function for individual component types enables a closed-form solution of the reliability of the system calculated as the product of the individual component reliabilities. Systems with parallel elements could also be modeled with modifications. The limitation of their reliability model is that it is not capable of accounting for consumable elements, the commonality of spares across component types, or system downtime. This limitation prevents the modeling of common elements of space system environmental control and life support systems and commonly configured functional primary/backup heterogeneous systems.

While the limitations of the system configurations Messinger and Shooman mean their models can't be used, their insight on the use of the knapsack problem should not be overlooked. Not only is the conceptual problem a direct analog, but it is also a popular discrete mathematics problem with several variations that can be altered to fit the problem at hand. In this work, a modified version of the bounded knapsack problem is implemented for optimization and suboptimization in the simulation and analytical approaches, respectively.

1.1.4 Including Commonality

On the other hand, approaches have also been developed in the literature to analyze the value of spare commonality. For example, Siddiqi and de Weck [17] developed a model that can consider shared common spares among different elements. The commonality model can be extended to cannibalize units from non-operating elements to expand the number of available spares in times of high need. This approach is accomplished through predefined operation timelines which establish when certain elements are operating or not to determine the spare availability. Although this method is effective at comparing the efficacy of increasing commonality within a given system, it lacks an inherent capability to allocate spares optimally, which is critical for mass-constrained systems like space habitats. This critical limitation of the traditional commonality-based approaches makes them inapplicable for the applications of interest to us.

More recently, Owens developed another approach that further investigates the impacts of commonality including distributed functionality and additive manufacturing on crew time [18]. The model used is similar to the one used by Messinger and Shooman [16]

and Siqqi and de Weck [17] but with a number of additional modeling parameters to account for the various supportability strategies that are evaluated and optimized as part of the multi-objective constrained optimization. However, even though the number of options along with the calculation of crew time requirements and the inclusion of epistemic uncertainty in this model represents the most flexible analytical approach available, the model is not able to account for a sparing approach that does not immediately initiate a repair action in the event of failure and only addresses internal redundancy at the component level, not the string level.

In this work, commonality is not treated as a special feature to be added. Rather, as part of defining the spares that can be paired with given components, commonality can be expressed by mapping similar components to the same spare.

1.1.5 Sparing Policies

While techniques have been proposed to estimate the spares allocations of conceptual crewed space systems, few have considered active choices of when to spare, what will be called the sparing policy of a system. This dimension of investigation is often foreclosed on because serial system reliability models require immediate sparing to bring the system back to a functional state. Such a policy, immediate repair-on-failure, should result in conservative estimates for the number of spares required but will fail to utilize internal redundancy, common in space systems, in the most efficient manner. For example, if a valve fails and brings down a string, repair-on-failure would say to replace the valve no matter the larger context of system. If there is a cold spare for the string and the replacement valve is also the spare for a valve supporting a single string elsewhere in the

system, repair on failure does not address whether using the spare valve helps to achieve the broader mission success goals rather it is provisioned without question. Given the mass impacts throughout the system of each additional kg of mass provisioned, an overly conservative estimate of the spares required may result in an unfair evaluation of a conceptual system even if it is able to meet the desired reliability; alternatively, other sparing policies that potentially delay repairs while leveraging backup systems can be more desirable for certain missions. Thus, there is a need for a rigorous and efficient approach to analyzing and optimizing sparing policies.

This work develops methods to include sparing policy investigations in both simulation and analytical frameworks. In the simulation-based solution, the EMAT is modified to test the repair-on-failure approach, termed the baseline policy, against a policy that delayed repairs until the backup system was necessary to continue to provide the CO₂ removal function. This new policy is called the lazy policy. This work demonstrates that policy choice impacts the estimated mass of spares required which would ripple through the overall set of systems by many orders of magnitude given the gear ratios involved. For the analytical framework, the parameters defining random variables representing repair and buffer depletion actions are identified as the underlying variables in a spare policy. By optimizing these inputs, the optimal strategy for a desired spares allocation mass, also called a target mass, can be found.

1.2 Research Objective

The objective of this research is to develop methods capable of including sparing policy along with system configuration in the estimation of functional availability and

associated spares allocations. For this research, sparing policy is the set of rules a crew uses to decide when to initiate a repair following a failure of a component. Sparing policy is often overlooked in system architecting and development phases due to the requirement for more detailed system configuration information to model the behaviors in question. However, sparing policy is a valid operational mitigation for uncertainty in component failure rates and time to repair and can impact the number of spares allocated to achieve the desired crew survivability targets. Multiple methods will be developed to look at sparing policy to meet different ends. The simulation-based approach will leverage existing tools, and it allows a granular level of control when desired to test a targeted sparing policy. The analytical method allows sparing policies to be explored and for an optimum policy to be found. In sum, the methods will meet the objective of this research by enabling robust exploration of sparing policy for crewed missions through the system design cycle.

To fully explore the research and how it meets the objective, the dissertation is organized in the following way: Chapter 2 introduces the details of the motivating application case study. Chapter 3 develops a simulation-based approach for testing specific spares policies and compares two different policies to have allocations optimized for each mass step. Chapter 4 presents an analytical method for optimizing spares policy at a specified mass. Chapter 5 summarizes the findings and suggests follow-on avenues for investigation.

CHAPTER 2. MOTIVATING CASE STUDY

The aim of this work is the development of the methods to explore different sparing policies to achieve the optimal tradeoff between the probability sufficiency and the spares mass for realistic systems. In general, the architecture of the types of systems typically used for long-duration crewed missions is a primary system with some level of internal redundancy supported by a heterogeneous backup system that utilizes a consumable element.

To model a realistic system and associated technologies, the systems that provide the carbon dioxide removal function of ISS are modeled and used as a test case for the developed methods. The Carbon Dioxide Removal Assembly (CDRA) and Lithium Hydroxide (LiOH) backup systems have been used in previous work [10] and represent the current state of the art performing a key function for any crewed ECLS system. If both systems fail, the percentage of CO₂ in the habitat will eventually grow to a point that is fatal to the crew and because it is a product of the crew respiration there is no way to capture and eliminate preferentially in the event of those systems failing. The fault trees used to describe how both systems fail is same as the one used in the previous work that utilized the Exploration Maintainability Analysis Tool (EMAT) [10]. The complexity of this case is in its redundant strings and consumable components; this complexity makes the traditional greedy spare strategy with a repair-on-failure policy inefficient and ineffective as shown later, which thus motivates developing the proposed modeling capabilities.

The following introduces the details of the considered application case study.

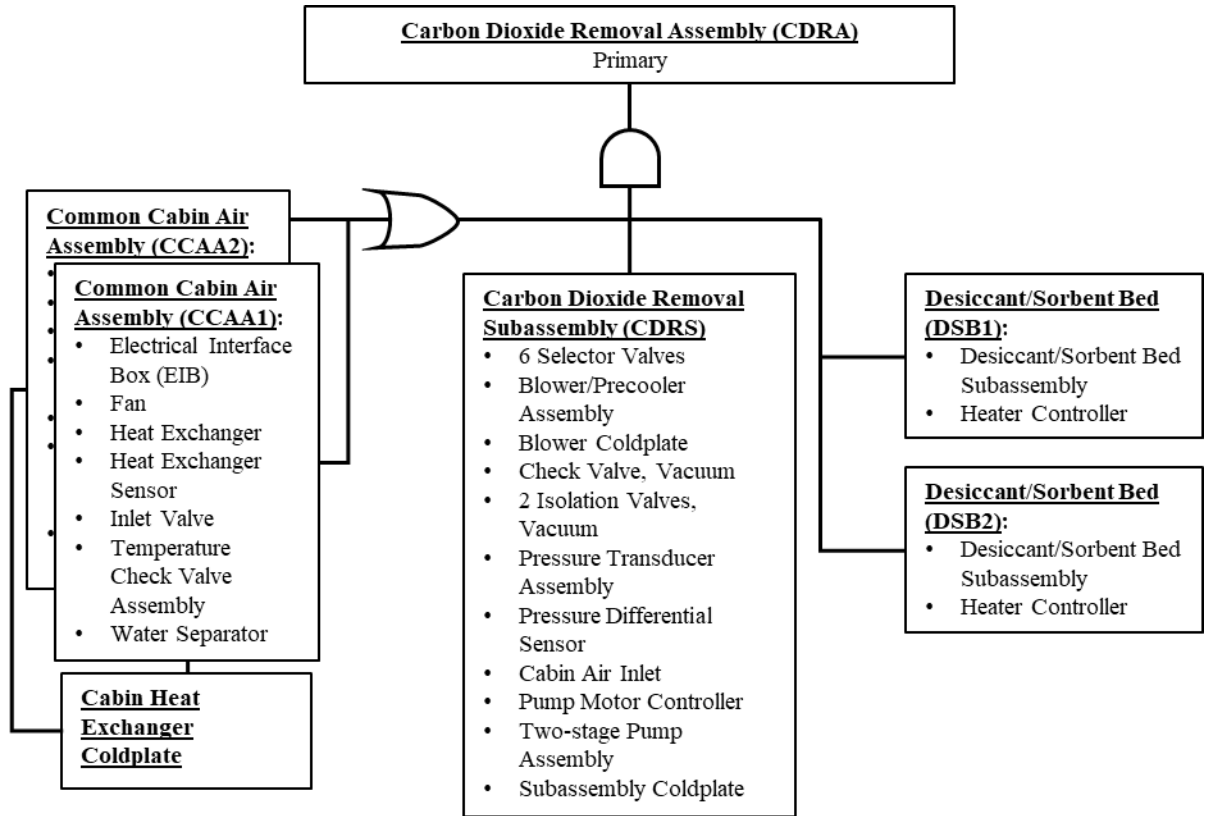


Figure 1: Carbon Dioxide Removal Assembly (CDRA)

The configuration of the primary system, the CDRA, is represented in Figure 1 grouped by the three main strings, the Common Cabin Air Assembly (CCAA), Carbon Dioxide Removal Subassembly, and the Desiccant/Sorbent Beds (DSBs) with the cabin heat exchanger coldplate existing as a one component string. Below the name of the string, all the components that comprise it are listed, and the key parameters of these components are captured in Table 1, which come from version 2.5 of the ISS Probabilistic Risk Assessment, an unpublished working model. In this table, there are 41 distinct components but only 20 distinct spare types. Each component of a string must be operating for the string as a whole to be operational. There are two DSB strings, both of which must be operating for the CDRA to function since one string is always capturing carbon dioxide while the

other is venting the gas to space. This need for two operational strings stands in contrast to the CCAA which only requires one to be fully functional while the other remains in reserve as a cold spare. The CCAAs are also unique in that they share one coldplate component that supports the heat exchangers for both strings. Being the primary system, any functional configuration of the system is preferred in the operation of the carbon dioxide removal function.

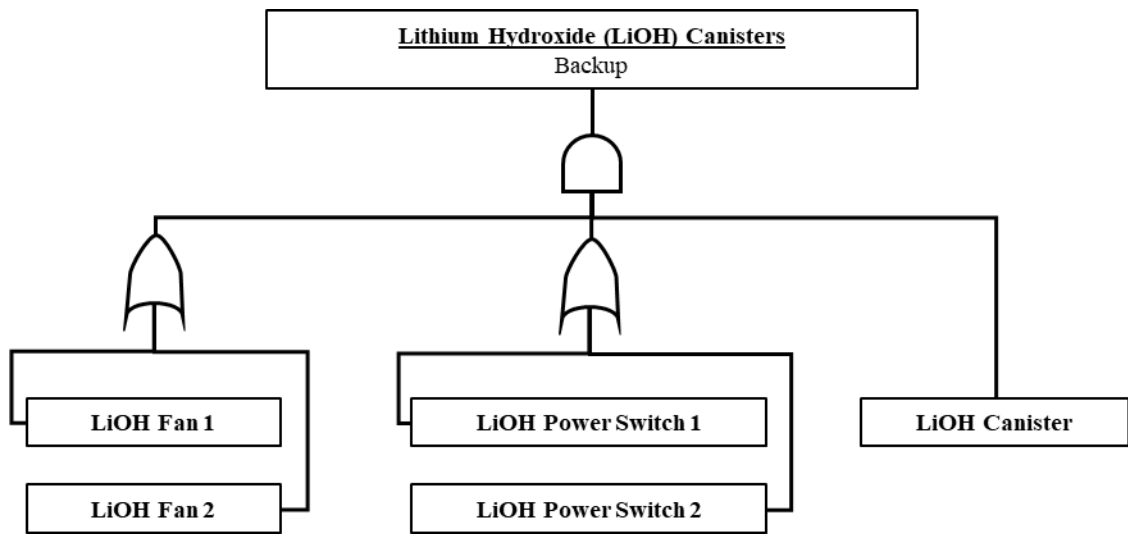


Figure 2: Lithium Hydroxide (LiOH) canister

Figure 2 shows the configuration of the backup system which is a simple system centered on the LiOH canister. This consumable element operates similarly to the DSB in that it contains a material that captures carbon dioxide gas. However, instead of then venting it to space, the canister is removed when saturated and discarded. The rest of the system is composed of cross-strapped fans and power switches to ensure there is a mechanism to push the cabin air over the canister. These components are all that comprise the back-up system and their parameters are also captured in Table 1.

Table 1: Component Parameters

| Name | State at time=0 | Failure Rate | Associated System | Spare type |
|---|------------------------|---------------------|--------------------------|---|
| CCAA1 Electrical Interface Box | 1 | 1.90E-05 | Primary | Electrical Interface Box (EIB) |
| CCAA1 Fan | 1 | 2.09E-05 | Primary | Fan |
| CCAA1 Heat Exchanger | 1 | 2.88E-05 | Primary | Heat Exchanger, Cabin Air, |
| CCAA1 Heat Exchanger Sensor | 1 | 6.38E-07 | Primary | Sensor, Temperature |
| CCAA1 Inlet Valve | 1 | 3.64E-05 | Primary | Inlet, Cabin Air |
| CCAA1 Temperature Check Valve Assembly | 1 | 7.68E-05 | Primary | Check Valve Assembly, Temperature Control |
| CCAA1 Water Separator | 1 | 6.87E-05 | Primary | Separator Water, orbital replacement unit (ORU) |
| CCAA1 Electrical Interface Box | 0 | 1.90E-05 | Primary | Electrical Interface Box (EIB) |
| CCAA2 Fan | 0 | 2.09E-05 | Primary | Fan |
| CCAA2 Heat Exchanger | 0 | 2.88E-05 | Primary | Heat Exchanger, Cabin Air-ORU |
| CCAA2 Heat Exchanger Sensor | 0 | 6.38E-07 | Primary | Sensor, Temperature |
| CCAA2 Inlet Valve | 0 | 3.64E-05 | Primary | Inlet, Cabin Air |
| CCAA2 Temperature Check Valve Assembly | 0 | 7.68E-05 | Primary | Check Valve Assembly, Temperature Control |

| | | | | |
|--|---|----------|---------|---|
| CCAA2 Water Separator | 0 | 6.87E-05 | Primary | Separator Water, ORU |
| CDRS Selector Valve 1 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Selector Valve 2 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Selector Valve 3 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Selector Valve 4 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Selector Valve 5 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Selector Valve 6 | 1 | 8.28E-05 | Primary | Valve, Selector, Assembly |
| CDRS Blower/Precooler | 1 | 0.000185 | Primary | Blower/precooler Assembly, CDRA |
| CDRS Blower/Precooler Coldplate | 1 | 6.67E-06 | Primary | Coldplate |
| CDRS Check Valve | 1 | 1.34E-05 | Primary | Valve Assembly, 1 in, Vacuum System |
| CDRS Isolation Valve 1 | 1 | 9.65E-06 | Primary | Valve, Manual, Isolation |
| CDRS Isolation Valve 2 | 1 | 9.65E-06 | Primary | Valve, Manual, Isolation |
| CDRS Pressure Transducer Assembly | 1 | 3.52E-05 | Primary | Pressure Transducer Assembly, Vacuum System |
| DSB1 Subassembly | 1 | 1.38E-04 | Primary | Desiccant/Absorbent Bed Assembly |
| DSB2 Subassembly | 1 | 1.38E-04 | Primary | Desiccant/Absorbent Bed Assembly |

| | | | | |
|--|---|----------|------------------------|------------------------------|
| CDRS Pressure Differential Sensor | 1 | 1.92E-05 | Primary | Pressure Sensor Differential |
| DSB1 Heater Controller | 1 | 9.89E-05 | Primary | Controller, Heater |
| DSB2 Heater Controller | 1 | 9.89E-05 | Primary | Controller, Heater |
| CDRS Cabin Air Inlet | 1 | 3.64E-05 | Primary | Inlet, Cabin Air |
| CDRS Pump Motor Controller | 1 | 1.06E-05 | Primary | Controller, Pump Motor |
| CDRS Pump Assembly | 1 | 1.54E-04 | Primary | Pump Assembly, Two-stage |
| Cabin Heat Exchanger Coldplate | 1 | 6.67E-06 | Primary | Coldplate |
| CDRS Coldplate | 1 | 6.67E-06 | Primary | Coldplate |
| LiOH Fan 1 | 0 | 2.09E-05 | Backup | Fan |
| LiOH Fan 2 | 0 | 2.09E-05 | Backup | Fan |
| LiOH Power Switch 1 | 0 | 2.82E-05 | Backup | CO2 Power Switch |
| LiOH Power Switch 2 | 0 | 2.82E-05 | Backup | CO2 Power Switch |
| LiOH Canister | 0 | N/A | Backup (Consumable) | Canister (1/day) |

For the simulation-based approach, the configuration and parameters are used as described, but for the analytical model a small simplifying choice is made. Specifically, for the LiOH backup system, the one-per-day consumption rate of canisters is maintained but the other non-consumable components, cross-straped fans and switches, are not

considered as part of the buffer depletion event. This makes modeling the potential for buffer depletion more straight forward and the omitted components are unlikely to contribute to failure. This is because the individual fans and switches have a mean time to fail on the order of 50,000 days compared to less than 30 days of operation and are cross-strapped so a failure of any one component does not result in failure of the backup.

CHAPTER 3. A SIMULATION APPROACH FOR TESTING SPARING POLICIES

This chapter discusses a state-of-the-art simulation tool, the Exploration Maintainability Analysis Tool (EMAT), and then presents two improvements. These extensions, an optimization method and a failure queue to implement spare policy testing, improve the allocations generated and the flexibility of the tool, respectively. The results also demonstrate the utility of the added context of an explicit spare policy and the associated impacts on estimated probability of sufficiency and spares allocations. Two policies are then compared across a range of scenarios that modify elements of the case study to investigate the impact of the policy differences compared to the parameter changes.

3.1 Simulation Methods

Overall, the simulation framework uses the results from a discrete event simulation with a general optimization framework for spare allocation enabled by the knapsack problem to evaluate various spare strategies. Optimal spares allocation is enabled through a dynamic programming solution to a modified version of the bounded knapsack problem utilizing the probability of sufficiency relative to the overall Monte Carlo results as a means of estimating the benefit of a component. Furthermore, EMAT is extended through the implementation of a repair queue that allows different rules-based sparing policies to be modeled and the results compared.

3.1.1 Existing EMAT

The Exploration Maintainability Analysis Tool (EMAT) analyzes the probability of sufficiency of a given spares allocation and uses a greedy optimization algorithm to generate the Pareto frontier between the probability of sufficiency and the spares mass. It relies on the activity scanning approach for discrete event simulation to join systems represented by continuous models with the discrete phenomena of interest. This implementation requires the discretization of time, effectively creating an opportunity for a standard set of events to change the state of the system. These events are driven by changes internal to the system as described by attributes of the entities of the model. Observing events and their subsequent impacts is the primary mode of operation in the activity scanning approach to discrete event simulation and is facilitated by the time step.

The overall flow of a given time step for EMAT can be seen in Figure 3. These elements are foundational to achieving the behavior desired from any implementation of the dynamic model. The three basic events of the tool are component failure, component repair, and consumable depletion.

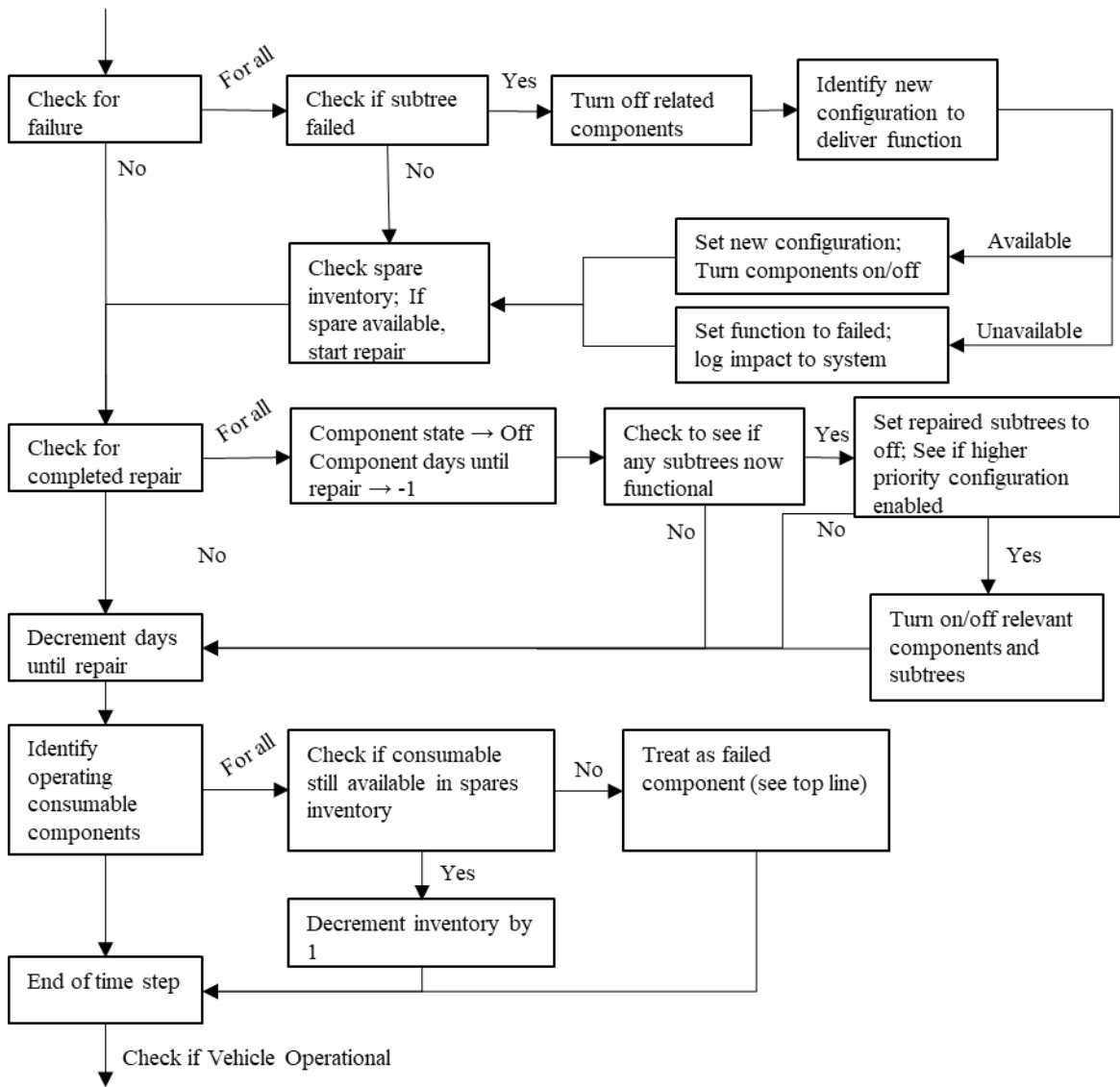


Figure 3: Time step flow of the EMAT tool

Previously, EMAT used a Greedy Algorithm to generate successive allocations to find the Pareto frontier between the probability of sufficiency and the spares mass; the same algorithm can be also used to find the optimal probability of sufficiency given the spares mass capacity, or the optimal spares mass given the probability of sufficiency requirement. This approach relies on adding the most “efficient” spare next. A formalized

version of the specific approach can be found in Sherbrooke [19] though variations can be found across optimal sparing literature [16,20]. For this implementation, that means running a set number of cases at the current allocation and observing how many times the lack of each spare may have contributed to the failure of a mission. These counts are then divided by the spares' respective masses and the spare with the highest count to mass ratio is added to the allocation; the process is repeated until a set mass capacity requirement is reached. In the previous version of EMAT, the consumables for the backup system were excluded from the process of determining the next spare to add because in every failed case the consumables are exhausted. To establish a fair comparison with the modified knapsack approach introduced later, a new rule in the Greedy Algorithm is implemented that adds a consumable if it is the only contributor to a failed mission. In the context of the problem, this condition corresponds to situations where a repair is in progress, but the consumable is exhausted prior to completion of the repair action. While the process is started with no spares in the allocation, a large mass capacity is set so that the Greedy Algorithm is used multiple times and the probability of sufficiency is evaluated for various cases of the spares mass below that capacity, which enables us to explore the entire Pareto front between the probability of sufficiency and the spares mass.

The pseudocode for the overall implementation is shown below.

Greedy Algorithm Pseudo Code

Input: Vehicle Configuration; Mission Duration T ; Mass of Each Spare m ($1 \times N$)

Parameters: iteration parameter $iterblock$; Capacity c [a large number to generate a Pareto front]

Initialize: $res = []$; $ite = 0$; $POS_{Pse} = []$; $M_{opt} = []$; Spares Inventory Quantity $spareinv_{initial}$, a $1 \times N$ sized vector

Repeat

Step 1 Record the number of iterations

$$ite = ite + 1$$

Step 2 For $i = 1$ to $iterblock$

Step 2.1 Simulate the mission for T with $spareinv_{initial}$ and the Vehicle Configuration. In the simulation, an additional rule is applied that sets $simoutput$ to 1 at the index corresponding to the consumable canister if, and only if, a repair is in progress and the consumable is exhausted.

Step 2.2 Store the simulation output vector identifying any spares that correspond to failed components not under repair at the end of the mission, a $1 \times N$ sized vector.

$$res[i, :] = simoutput$$

End

Step 3 Calculate the number of successful mission instances

$$SMiss = count(sum(res, 2) = 0)$$

Step 4 Calculate the number of times each spare contributed to a failed mission

$$SpareFailCount = sum(res, 1)$$

Step 5 Calculate Spare Inventory Probability of Sufficiency and store; an $ite \times 1$ sized vector

$$POS_{Pse} = \left[POS_{Pse}; \frac{SMiss}{iterblock} \right]$$

Step 6 Calculate Spare Improvement Metric for Each Spare (element-wise division); a $1 \times N$ sized vector

$$SpareImprovementMetric_j = \frac{SpareFailCount_j}{m_j} \quad \forall j \in [1, N]$$

Step 7 Determine Spare with Maximum Improvement Metric (index j^*)

Step 8 Add one to Spares Inventory Quantity of Spare Identified in Step 6 and store new allocation

$$spareinv_{initial}[j^*] = spareinv_{initial}[j^*] + 1$$

Step 9 Multiply $spareinv_{initial}$ history by spares mass to get corresponding mass for each calculated Pos , an $ite \times 1$ sized vector

$$M_{Pse} = [M_{Pse}; spareinv_{initial} \cdot m^T]$$

Go to Step 1**Until** $\text{spareinv}_{\text{initial}} \cdot m^T \geq c$ **Output** $M_{Pse}, PoSPse$

There are a few limitations in EMAT that prevent it from being applicable in this application.

First, while the implementation of this Greedy Algorithm is relatively simple, there is a concern when the spares mass spans a large range (e.g., from 0.5kg to 120kg in this case). In such a situation, the Spare Improvement Metric (see Step 6 in the pseudocode above) may overly favor low mass spares at high probabilities of sufficiency because few failures are being generated due to the finite length of the simulation time horizon. Because of this, using the Greedy Algorithm could lead to suboptimal allocations or requirements that do not mesh well with the constrained environments of long-duration human exploration missions. In this work, a new algorithm based on a modified solution to the Knapsack Problem is developed to optimize the spare allocation efficiently.

Second, a close read of the flow of the time step in Figure 3 indicates that if a spare is available for repair, the repair action is initiated. This approach is an implicit implementation of a sparing strategy, repair-on-demand. However, this policy would do a disservice to systems with a high level of internal redundancy in terms of the probability of sufficiency because it will require a spare even if the system can continue to operate on a different string. In this work, a spare queue is implemented within the vehicle and time step to enable different sparing policies to be tested.

The subsequent subsections introduce each of these improvements to the EMAT tool.

3.1.2 Optimal Spares Allocation

First, a new way of determining the optimal spares allocation is implemented. Rather than the Greedy Algorithm approach of initializing a minimal spares allocation, running the Monte Carlo, determining best spare to add, and repeating, the algorithm is divided into two parts: simulations (i.e., mission instances generator) and optimization (i.e., Modified Knapsack Problem). First, the simulation is iterated many times, on the order of hundreds of thousands, to generate a representative set of spare demands for successful missions. The optimization routine then uses a recursive dynamic programming approach comparing the benefit of the previously observed best against candidate allocations available at given mass capacity.

The simulation is used to create a set of data used for optimization. First, the spares inventory is initialized with an unrealistically high quantity, 10^6 , for each spare type. With an initialized system that can be simulated over many missions, the system is simulated until the convergence criteria are met. At the end of each mission instance, the difference between the initial and final spares inventory is computed resulting in the minimum number of spares that are sufficient for that specific mission to be successful. The result is a large set of successful missions and the requisite sparing allocations necessary to enable them. The pseudocode for this step in the evaluation is presented below:

Mission Instances Generator Pseudo Code

Input: Vehicle Configuration; Mission Duration T

Parameters: iteration parameter $iterblock$; convergence parameter $convper$;

Initialize: $res = []$; $spareinv_{initial} = T * f * ones(1, N)$ where f is a large integer constant; $TotalSims=0$

Repeat

Step 1 For $i = 1$ to $iterblock$

Step 1.1 Simulate the mission for T with the initialized Vehicle configuration

Step 1.2 Calculate the number of spares required to complete the simulate mission a $1 \times N$ sized vector

$$res[i + TotalSims, :] = spareinv_{initial} - spareinv_{final}$$

End

Step 2 Calculate the mean quantity of each spare for res before and after the current block, $1 \times N$ sized vectors:

$$\overline{res}_{initial} = mean(res[1:TotalSims, :], 1)$$

$$\overline{res}_{final} = mean(res, 1)$$

Step 3 Determine element-wise percentage difference between the initial and final means and determine the maximum value

$$\overline{res}_{max} = max\left(abs\left(\frac{\overline{res}_{final} - \overline{res}_{initial}}{\overline{res}_{final}} \right) \right)$$

Step 4 Update the total number of simulations

$$TotalSims = TotalSims + iterblock$$

Until $\overline{res}_{max} < convper$

Output: $TotalSims, res$ (a $TotalSims \times N$ sized matrix)

With a large number of missions simulated, they can be used to determine the spare allocations that provide the most benefit in the area of probability of sufficiency. Again, this measure is essentially the probability that the spares allocated are sufficient to meet the needs of the mission. The goal is to identify the Pareto front between the probability of sufficiency, $PO_{SMKP,S}$, and the spares mass M_{opt} . By alternatively viewing this problem as a single-objective optimization for $PO_{SMKP,S}$, the goal can be rewritten as optimizing

$Pos_{MKP,S}$ given a constraint on the capacity c . Namely, $Pos_{MKP,S}$ is modeled as a function of a set of spares, X , that are composed of a certain amount, x_s , of each spare type s . Denoting the individual masses of each spare s as m_s , this problem can be formulated as:

$$\text{maximize } Pos_{MKP,S}(X)$$

$$\text{subject to: } \sum_{s=1}^n x_s m_s \leq c$$

$$x_s \geq 0$$

(1)

The solution used for the knapsack problem in this case is an extension of the Bellman recursion [20] which leverages a dynamic programming approach to repeatedly solve the same subproblem, but with a deviation from the typical calculation of benefit. In normal solutions to a knapsack problem, the discrete value of a given item is known and the linear combination of items by their values is used to calculate the total value of the knapsack. In this case, the value of a given spare is not known prior to the optimization. Rather, the subproblem generates candidate allocations and these are compared to the set of successful mission allocations from the Simulation Step described in the Mission Instances Generator. Therefore, it is appropriate to call this problem a Modified Knapsack Problem (MKP). Equation 2 shows how the probability of sufficiency is calculated from the results of the simulation, Pos_{sim} .

$$PoS_{sim}(X) = \frac{\text{count}(\text{res}(:,j) \leq x_s \forall s)}{\text{TotalSims}} \quad (2)$$

where *count* is the function to count the number of rows of *res* that satisfy the condition $\text{res}(:,s) \leq x_s \forall s$.

In the recursive dynamic programming algorithm, each spare type *s* is added incrementally, and calculate the probability sufficiency $PoS_{MKP,S}(X_{d,j})$ that is the optimal probability of sufficiency given the (discretized) mass, *d*, using the spare types up to *s*. The goal is to populate the matrix table $PoS_{MKP,S}(X_{d,s})$ as well as the corresponding allocation $X_{d,s}$ for each entry. When each spare type *s* is added, candidate allocations, $X_{d,s}$, are generated with every possible number of spares, *l*, for the given mass, *d*. The corresponding *PoS* candidates are compared to the current best *PoS* and supplant it if any of them is greater; otherwise, the previous *PoS* is retained, and the next mass spare type *s+1* is tested. The subproblem used to generate test allocations and select the best is shown in Equation 3 below.

$$PoS_{MKP,S}(X_{d,s}) = \begin{cases} PoS_{MKP,S}(X_{d,s-1}) & \text{if } d < m_s \\ \max_{l=1, \dots, \text{floor} \frac{d}{m_s}} \{PoS_{MKP,S}(X_{d,s-1}), PoS_{sim}(X_{\text{floor}(d-l \cdot m_s), s-1} + l \cdot x_s)\} & \text{if } d \geq m_s \end{cases} \quad (3)$$

Equation 3 is implemented in the dynamic programming optimization described in the modified knapsack problem pseudocode below. $PoS_{MKP,S}(X_{d,s-1})$ is the best previously calculated probability of sufficiency for that mass step and is being updated from Equation (3), whereas the PoS_{sim} term must be determined from the simulation results using the candidate allocation and Equation 2. Just like in the Greedy Algorithm, the parameter

Capacity c can be set to a large value to output the Pareto front between PoS_{opt} and M_{opt} .

Using res from the Simulation Step (i.e., Mission Instances Generator), one example of

Capacity c can be specified as follows:

$$c = m \cdot \max(res)^T$$

Modified Knapsack Problem Optimizer Pseudo Code

Input: Results matrix from Mission Instances Generator res , Total simulations performed in Mission Instances Generator $TotalSims$, Mass of Each Spare m , a $1 \times N$ sized vector, Capacity c (a large number to generate a Pareto front)

Parameters: Mass Discretization $massdis$;

Step 1 Calculate the number of mass steps, D

$$D = \frac{c}{massdis}$$

Step 2 Calculate the Probability of Sufficiency with no spares; determines the expected PoS if no spares were provisioned by calculating the percentage of missions that did not require any spares

$$PoS_{no\ spares} = \frac{count(sum(res, 2) = 0)}{TotalSims}$$

Step 3 Initialize Probability of Sufficiency Matrix $posTable$, a $D+1 \times N+1$ sized matrix, and set its first row $posTable[1,:]$ and first column $posTable[:,1]$ to $PoS_{no\ spares}$

Step 4 Initialize Allocation Multidimensional Array $allocStorage$, a $D+1 \times N+1 \times N$ sized array, and set its first row $allocStorage[1,:,:]$ and first column $allocStorage[:,1,:]$ to an n -dimensional vector of zeros. This array stores the corresponding spare allocation for each case in $posTable$; $allocStorage[i,j,k]$ stores the number of spares allocated for spare of type k for $posTable[i,j]$.

Step 5. For each spare type $s = 1$ to N

Step 5.1 For each mass step $d = 1$ to D

Step 5.1.1 Setting $PoS(X_{d,s-1}) = posTable[d+1, s]$, solve Equation 3

$$PoS_{MKP,S}(X_{d,s}) = \begin{cases} PoS_{MKP,S}(X_{d,s-1}) & \text{if } d < m \\ \max_{l=1, \dots, \lfloor \frac{d}{m_s} \rfloor} \{PoS_{MKP,S}(X_{d,s-1}), PoS_{sim}(X_{\lfloor (d-l)m_s \rfloor, s-1} + l \cdot x_s)\} & \text{if } d \geq m \end{cases}$$

Here, the PoS_{sim} candidates are determined with the X input and evaluated using res via Equation 2.

Step 5.1.2 Store the PoS calculated in Step 5.1.1 in $posTable[d+1, s+1]$

Step 5.1.3 Store the X that results in the PoS from step 5.1.1 as $X_{d,s}$ in $allocStorage[d+1, s+1, :]$

End

End

Step 6 Calculate mass of optimal allocation at each step (shiftdim function reduces $D+1 \times 1 \times N$ multidimensional array into a $D+1 \times N$ sized matrix); results in a $D+1 \times 1$ sized vector

$$M_{opt} = \text{shiftdim}(allocStorage[:, N + 1, :]) * m^T$$

Step 7 Break out the last column of the Probability of Sufficiency matrix to pair with the mass vector calculated in Step 6, a $D+1 \times 1$ sized matrix

$$PoS_{opt} = posTable[:, N + 1]$$

Output: M_{opt}, PoS_{opt}

In this way, the simulation approach utilized enables a robust optimization approach based on dynamic programming that can identify the preferred allocation for a given mass to maximize the probability of sufficiency. This approach directly identifies both the candidate allocation and the probability of sufficiency ensuring that performance at that mass is achievable and eschewing intermediate benefit metrics that may lose efficacy depending on the actual configuration of the system. Additionally, the MKP approach can consider the consumables along with the rest of the spares since it is concerned with allocations that result in successful missions rather than attributing failure to unrepaired elements. The application of the optimization methodology to the selected simulation approach results in a powerful combination capable of identifying allocations that lead to an optimal tradeoff between the probability of sufficiency and the mass.

In Section 3.2, the MKP approach is compared to the Greedy Algorithm because the latter is the traditional method used with EMAT. Note that, although MKP is used as the optimizer because a dynamic programming solution to the knapsack problem is well-

suiting to the results of the updated EMAT model and the approach of generating a large number of successful missions, other approaches, such as branch-and-bound [11,20], Lagrange multipliers [16], or multi-echelon techniques [19], could be investigated in the future. If the problem statement for these optimizers can be aligned to work on a set of successful missions, they could even be used on the same results already created by the mission instances generator.

3.1.3 Implementing Testing of Sparing Policies with a Failure Queue

To implement the flexibility to test different repair policies, the discrete event simulation structure must be modified. Most importantly, the failure queue needs to be created so the order of repairs can be remembered across time steps. This construct allows the simulation to track what repairs are waiting to be completed and for the modeler to establish and test conditions under which a repair is initiated causing the failed component to leave the queue and a repair is subsequently completed. Instantiating a failure queue also expands the flexibility of the tool to leverage prioritization and value ranking of potential spare events. In practice, the queue is a tracking mechanism initialized as an empty matrix, assuming no components failed at the start of the mission, and then adds a failed component's id and spare inventory index as a row vector on failure. When a repair action for that item is initiated, the component is removed from the failure queue.

A modified version of the EMAT time step is presented here in Figure 4 and the key areas modified are identified with a number and dashed box around the impacted step. When a component fails, the requisite information must be identified and added to the queue in the first modified step (1). Then, like before, the simulation goes through the

necessary steps to reconfigure the system in the event the failure drives a change. The simulation performs a check at the modified step (2) to see if the predefined sparing policy conditions are met to move a component or components from the failure queue into repair. Because EMAT accounts for the number of time steps to repair, the item is not instantly repaired, but the process is initiated. The simulation then goes through the standard process before performing a second sparing policy-related check, at the modified step 3, at the end of the repair process. The second check gives the rule maker the option to initiate subsequent repairs in the queue based on the post-repair state of the system. The flow then continues as before (e.g., Figure 3).

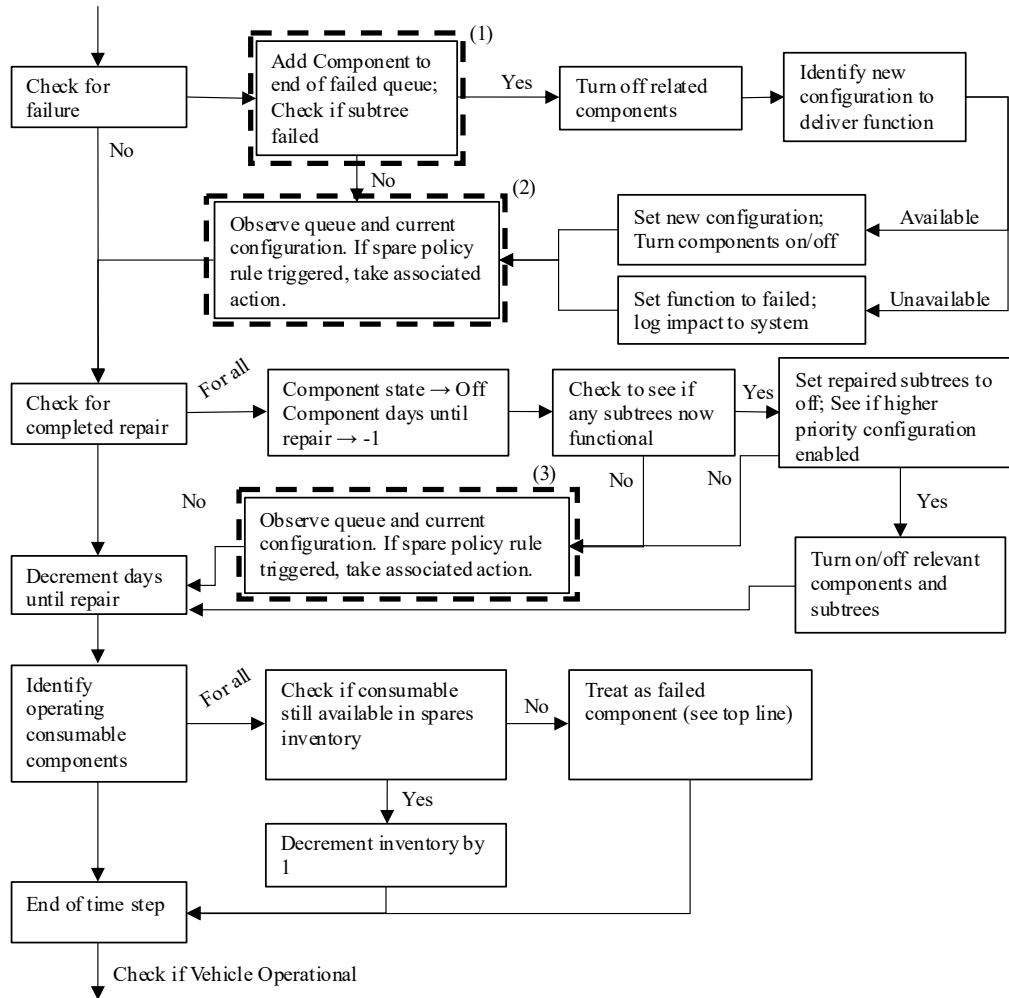


Figure 4: Time step flow modified for generic sparing policies

Note that implementing the original EMAT sparing policy (and the implicit approach of most of the other repair-on demand methods) with the new time step is trivial. As components fail, they are added to the queue as shown in the first modified step (1). At the end of the component fail set of actions, there is a check to see if there are any components in the queue and repair actions are initiated for all components in the queue (2). The spare countdown is started, and the process proceeds as before. The second check (3) is passed with no action since the queue will always be empty at that point in the step

since the queue is emptied because of the first check (2) and there are no opportunities for additional failures between the updated steps (2) and (3).

An interesting utilization of the new queue functionality is the Lazy Policy. In this policy, sparing is only initiated once the primary system sustains enough component failures so that it is no longer operational, and the backup is required. The first component to fail is then queued for repair. Once that repair is complete, the functionality of the primary system is checked. If this system can operate again, it is turned on and repairing stops. Otherwise, the next component in the queue, the component that failed second, is repaired until the primary is functional. This first-in, first-out (FIFO) approach does not allow for any prioritization of the queue, but this means the lazy crew does not have to make any judgment about what to repair next. In the simulation, components fail and are added to the queue (1). When the first check is reached (2), the simulation tests the priority of the system providing a given spacecraft function. If it is the lowest, thus the backup, then repair of the component at the top of the queue is initiated. The time steps continue with the system functioning on the backup until sufficient time elapses for the repair to complete. The impact of the repair is propagated through the system. If the backup is still necessary, the second check (3) initiates a repair action for the component now at the top of the queue.

The simulation combined with the generalized queue allows for different sparing policies to be tested. Adding the queue and supporting check actions enables the functionality that even EMAT does not have. The performance of the baseline policy, repair-on-failure, compared to the new lazy policy in section 3.3.

3.2 Comparison of Optimization Approaches

In this section, the simulation is run for both the Greedy Algorithm and the MKP optimization approach introduced in Section 3.1.2 to compare their performance. The Greedy Algorithm's inventory begins empty and adds spares and consumables as determined by the algorithm. This is a departure from the approach of starting with some consumables initialized utilized in previous EMAT work with the Greedy Algorithm and is done to provide a fair comparison with MKP.

The simulation parameters are selected to try to give as close of a comparison as possible. Previously, EMAT used 1,000 iterations per step to calculate the probability of sufficiency; however, it is known that a larger number of simulations would result in a more accurate result. Thus, in this comparison, the Greedy Algorithm is run with 1,000 and 50,000 simulations per step and stops simulating at 600kg. (Note that although a convergence criterion can be used to stop simulations when some threshold of stability is reached, the number of simulation steps is used as a convergence criterion for the two approaches for a fair comparison.) The MKP approach is also run with 1,000 and 50,000 total simulations so the results can be fairly compared with the Greedy Algorithm. The other parameters used for each approach are summarized in Table 2. The results of the two cases can be seen in Figure 5 including the time necessary to generate the results in MATLAB R2018b on a laptop with an Intel® Core™ i5-8350 CPU with 16GB of installed RAM.

Table 2: Optimization Approach Comparison Parameters

| Parameter | Value |
|-----------|-------|
|-----------|-------|

| Greedy Algorithm Parameters | |
|------------------------------|--|
| <i>iterblock</i> | 1,000 and 50,000 steps |
| <i>c</i> | 600 kg |
| Modified Knapsack Parameters | |
| <i>iterblock</i> | 1,000 and 50,000 steps |
| <i>c</i> | 600 kg |
| <i>convper</i> | 100% (forces only one iteration block) |
| <i>massdis</i> | 1 kg |

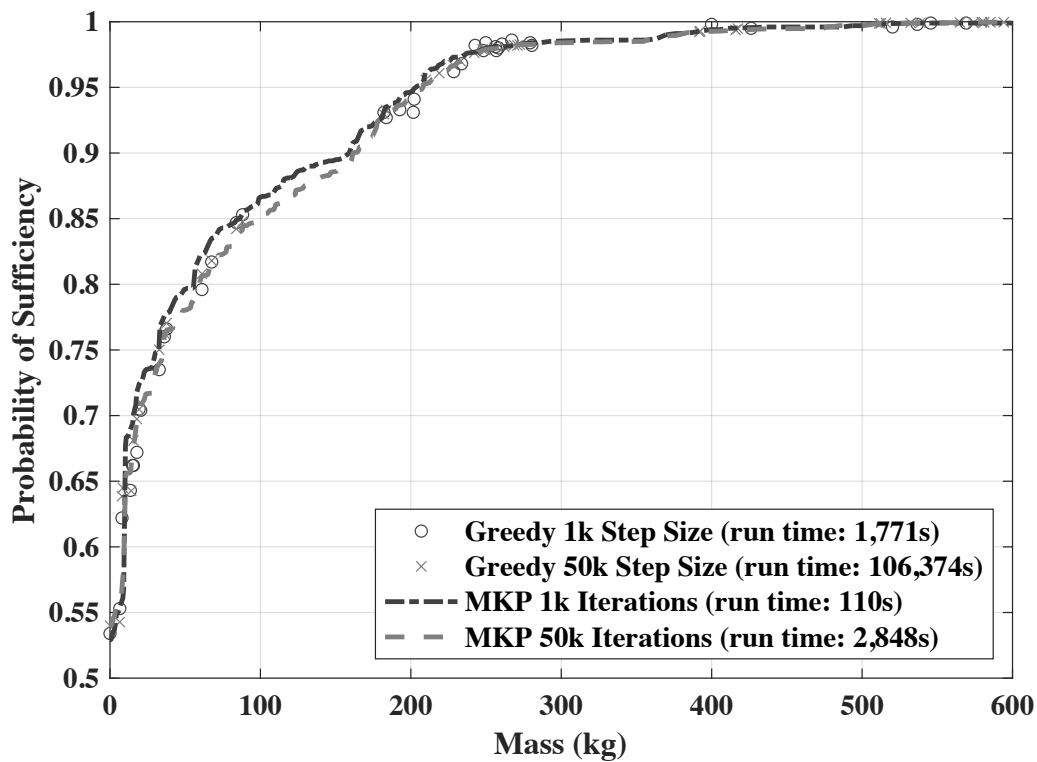


Figure 5: Comparison between the modified knapsack approach and the Greedy algorithm

The first finding from Figure 5 is that the results from 1,000 iterations per step do not match well with the results from 50,000 iterations per step. This shows that 1,000

iterations per step used in the traditional EMAT are insufficient for this application to achieve stable convergence. Therefore, for the rest of this part of the case study, 50,000 iterations per step for evaluation are used.

The major benefit provided by the MKP approach is reducing total run time without a commensurate reduction in performance. Because it does not need to run a large number of cases to stably calculate the probability of sufficiency at each mass step, the MKP optimization is able to more efficiently use the simulations run to determine the probability of sufficiency for all of the allocations. An improvement of one to two orders of magnitude in run time is achieved without sacrificing accuracy in tracing the Pareto frontier. Additionally, the MKP approach provides PoS discretized by kg, or any other mass step, which gives additional flexibility whereas the Greedy Algorithm will jump by the mass of the spare which could leave areas in-between unexplored. These attributes make the implementation of MKP within the broader EMAT framework a useful improvement over the Greedy Algorithm.

3.3 Comparison of Sparing Approaches

With the ability to test different sparing policies enabled through the modification of the time step approach in Section 3.1.3, the effectiveness of different policies can be tested as well.

3.3.1 Comparison Between the Baseline Policy and Lazy Policy

Using the same initialization state (e.g. system configuration, spares masses) and parameters (see Table 3), system operation is simulated over 360 days repeatedly for both

the repair-on-failure (baseline) sparing policy and Lazy Policy, the “lazy crew” sparing policy discussed in Section 3.1.3. The results of each set of simulations are then inspected using the dynamic programming optimizer to identify the allocation for each kg to optimize the probability of sufficiency. The results are presented in Figure 6.

Table 3: Sparing Policy Comparison Parameters

| Parameter | Value |
|------------------|--------------|
| <i>iterblock</i> | 50,000 steps |
| <i>convper</i> | 5% |
| <i>massdis</i> | 1 kg |

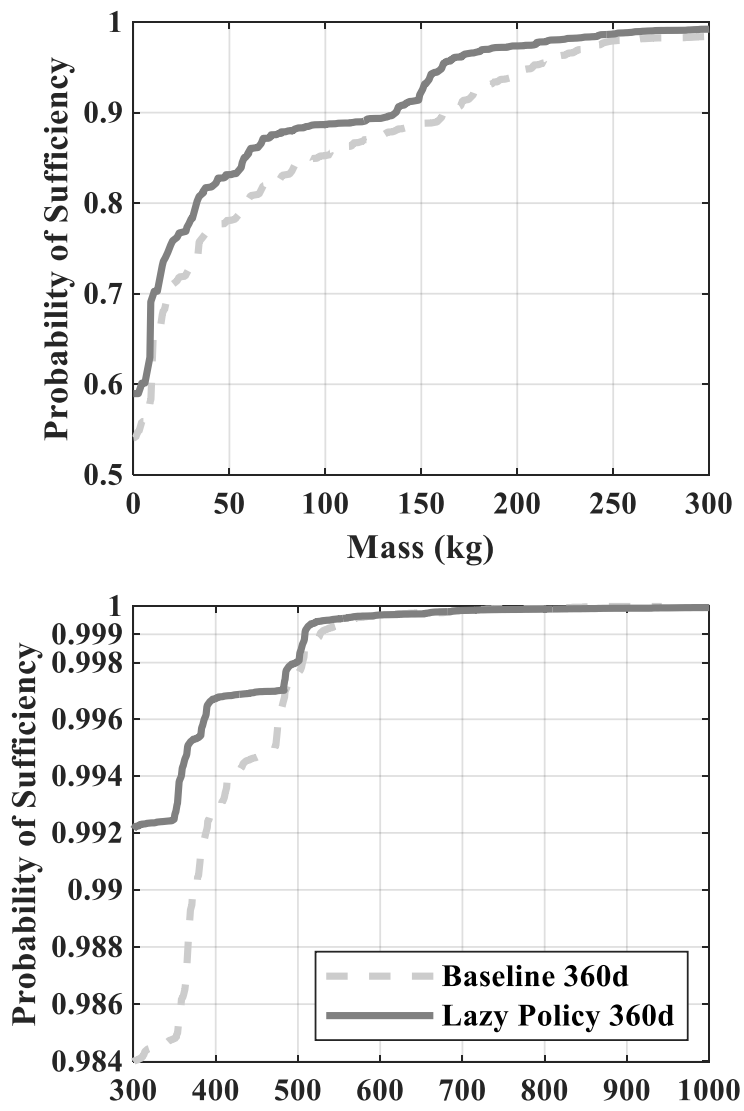


Figure 6: Comparison between the baseline and lazy policy

(Note: The bottom figure is a continuation of the top figure utilizing a log scale on the y-axis to improve visibility in a region of interest.)

Clearly, changing the sparing policy has an observable impact on the sparing mass required to achieve the desired function probability of sufficiency. In the zero spares mass case, there is an improvement of ~5% to the probability of sufficiency which represents the

subset of cases in the baseline approach that would have been spared even though the backup system would never have been utilized. As the spares mass increases, the Lazy Policy demand satisfaction dominates as expected. At 99.9% probability of sufficiency, however, the marginal spares mass between the two policies is less than 20kg (508kg v. 527kg) and the advantage deteriorates from there until there is a marginal difference between the policies. This result suggests that the number of spares required to achieve a very high probability of sufficiency is less sensitive to the spare strategy; rather, it is dominated by the component failure rates and configuration.

Table 4: Comparison of spares policies' allocation at 99.9% demand satisfaction

| Spare Name | Baseline Policy Spares | Lazy Policy Spares |
|--|-------------------------------|---------------------------|
| Valve, Selector, Assembly | 3 | 3 |
| Valve, Manual, Isolation | 2 | 1 |
| Valve Assembly, 1 in., Vacuum System | 1 | 1 |
| Separator Water, ORU | 1 | 1 |
| Sensor, Temperature | 1 | 0 |
| CO2 Power Switch | 0 | 0 |
| Pump Assembly, Two-stage | 2 | 2 |
| Pressure Transducer Assembly, Vacuum System | 1 | 1 |
| Pressure Sensor Differential | 2 | 1 |
| Inlet, Cabin Air | 2 | 1 |
| Heat Exchanger, Cabin Air-ORU | 1 | 1 |
| Fan | 1 | 1 |
| Electrical Interface Box (EIB) | 1 | 1 |
| Desiccant/Absorbent Bed Assembly | 2 | 2 |
| Controller, Pump Motor | 1 | 1 |
| Controller, Heater | 2 | 2 |
| Coldplate | 1 | 1 |
| Check Valve Assembly, Temperature Control | 2 | 1 |
| Canister | 7 | 8 |
| Blower/precooler Assembly, CDRA | 2 | 2 |

Table 4 shows the comparison of spares policies' allocation at 99.9% demand satisfaction. Looking at this table, the allocations between the two policies are very similar. In fact, 14 of the 20 spares, or 70%, have the same allocation. For five of the six spares with changes, each has one less in the Lazy Policy allocation accounting for the reduction in the masses between the two policies. Perhaps most surprisingly, the number of consumable canisters required only increases by one even though Lazy Policy explicitly seeks to delay making repairs until the backup system using this consumable is required. In this case, the more aggressive spare policy does not result in a large growth in required consumables.

As shown in Figure 6 and Table 4, changing the sparing policy has a clear impact on the baseline results. A higher risk approach to degraded performance results in more successful missions for a given spares mass especially at low spares masses. As the demand satisfaction increases, this benefit wanes.

3.3.2 Impact of Sparing Policy Changes on Alternative Configurations

Having established the impact of varying the sparing policy on the baseline configuration, it is worth evaluating its impact on other configurations, such as the cases with different repair times, different numbers of strings, and different failure rates. This is done to test the contention that changing the spare policy can be used as a means to reduce the spares mass or increase the probability of sufficiency with minimum impact on developmental effort and cost compared to other alternative approaches. The tradeoff may be in the operational effort, cost, or complexity but such a trade may become appealing if the

budgets are limited or the demonstrated capability does not meet the anticipated or allocated performance.

3.3.2.1 Days to Repair

One parameter that may affect the impact of the spare policies is the number of days to repair. In case the repair time is longer than planned, changing the spare policy may be a valid approach to mitigate the broader system impact of the underperformance. Shifting from the baseline repair-on-failure approach to the Lazy Policy may exacerbate issues with increased days to repair but it is worth testing to see the impact. Given that the Lazy Policy drives the system to create a queue that addresses repairs one at a time after the backup system is engaged, it is reasonable to expect that the impact of different days to repair to be more pronounced, especially for the increased days to repair case.

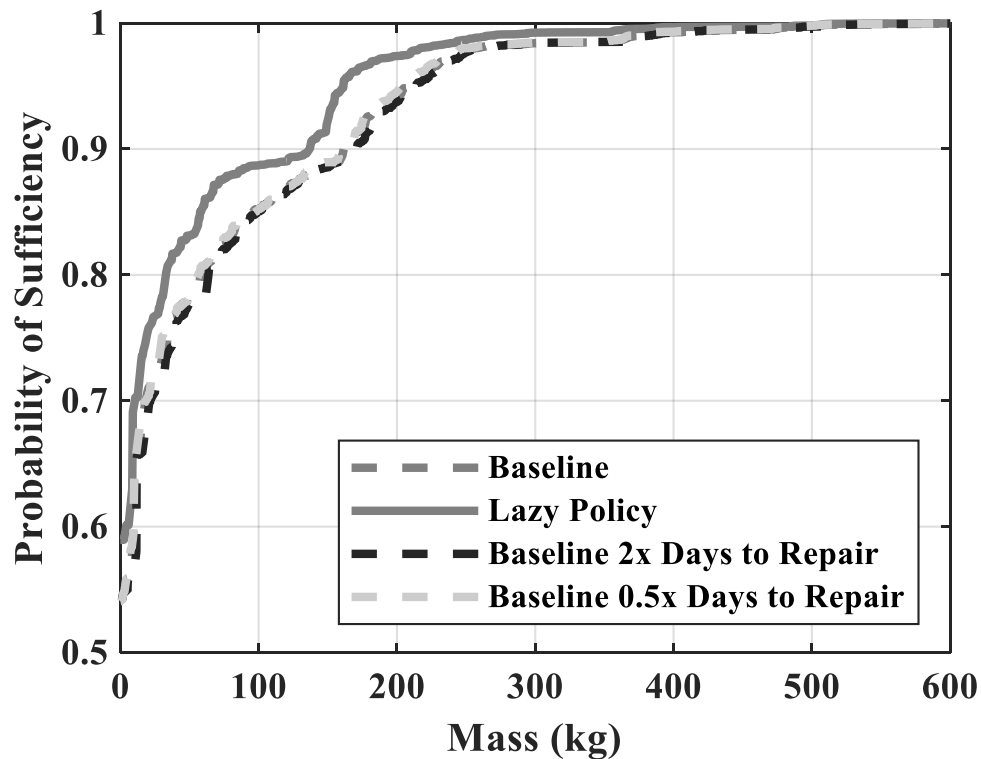


Figure 7: Alternative days to repair results compared

In reality, the impact of changing the days to repair is minor as shown in Figure 7 compared to the benefit gained by switching between policies. Comparing the results of the alternatives using the baseline repair-upon-failure policy and the regular days to repair with the Lazy Policy, it shows that the latter can provide a similar or better probability of sufficiency at a given mass. This result suggests that, for the considered system, investing additional resources to improve or maintain the number of days to repair would not be very impactful compared with changing the sparing policies, the latter of which requires minimum engineering effort to implement.

However, if the repair times grew by a factor of five, as shown in Figure 8, rather than two, its impact could be more significant. This case can happen when the placement

of a system made it hard to access, or the design of the system made the diagnoses and repair difficult for particular components. To test such a case, an additional case with the days to repair increased by five times (5x) with the Lazy Policy is simulated, and Figure 8 shows the results compared to the baseline case and Lazy Policy case. The degree to which the 5x case for the Lazy Policy approximates the baseline results is rather stunning. Essentially, all the gains provided by the change in sparing policy are given back if the system requires significantly more time to diagnose and repair. Conversely, if a program began with the Baseline Policy and found that the system required significantly more time to repair resulting in a lower probability of sufficiency for the spared mass, the switch could be made to the Lazy Policy to operationally mitigate the reduced performance without investing resources to improve the repairability of the components. The overall efficacy of such a switch will need to be modeled since a switch in policy is not sufficient to overcome any parameter change, as shown below in Section 3.3.2.3.

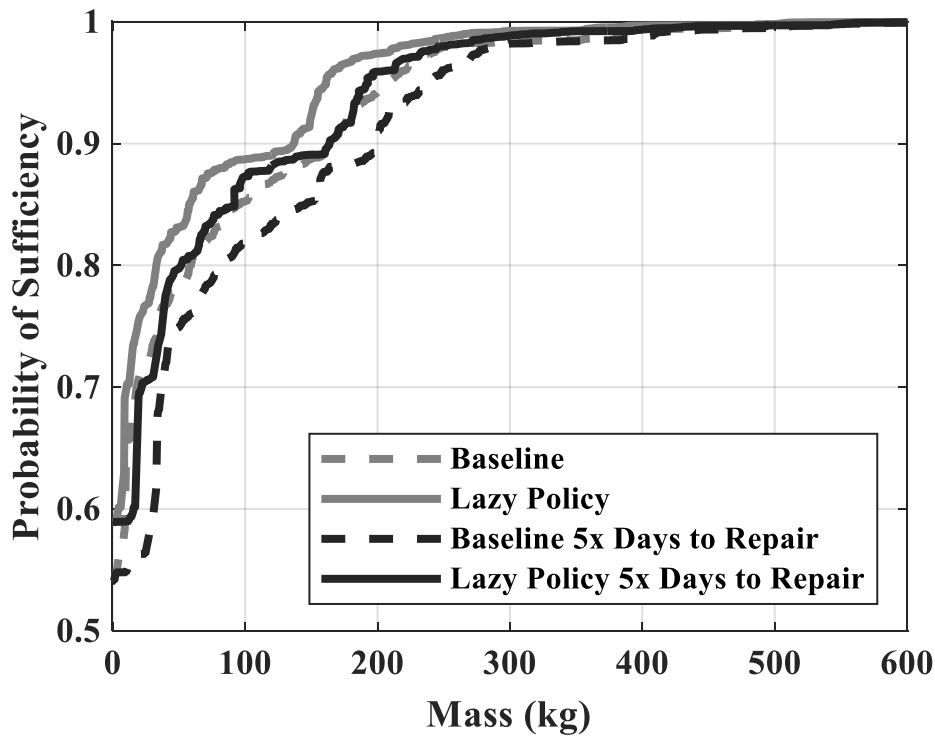


Figure 8: Lazy policy with 5 times days to repair

3.3.2.2 Additional Strings

This subsection evaluates the impact of changing the sparing policy on the alternative configurations with different number of strings because they have direct impact on spare demand. This approach is tested by adding an additional desiccant/sorbent bed string and an additional common cabin air assembly (CCAA) string to the model and repeating the analysis. This result is shown in Figure 9. Note that there is an initial mass penalty associated with including additional strings; therefore, the case with the additional strings requires more mass than the baseline case. The focus here is to observe how the impact of the Lazy Policy compared to the Baseline Policy would differ in each of these two cases.

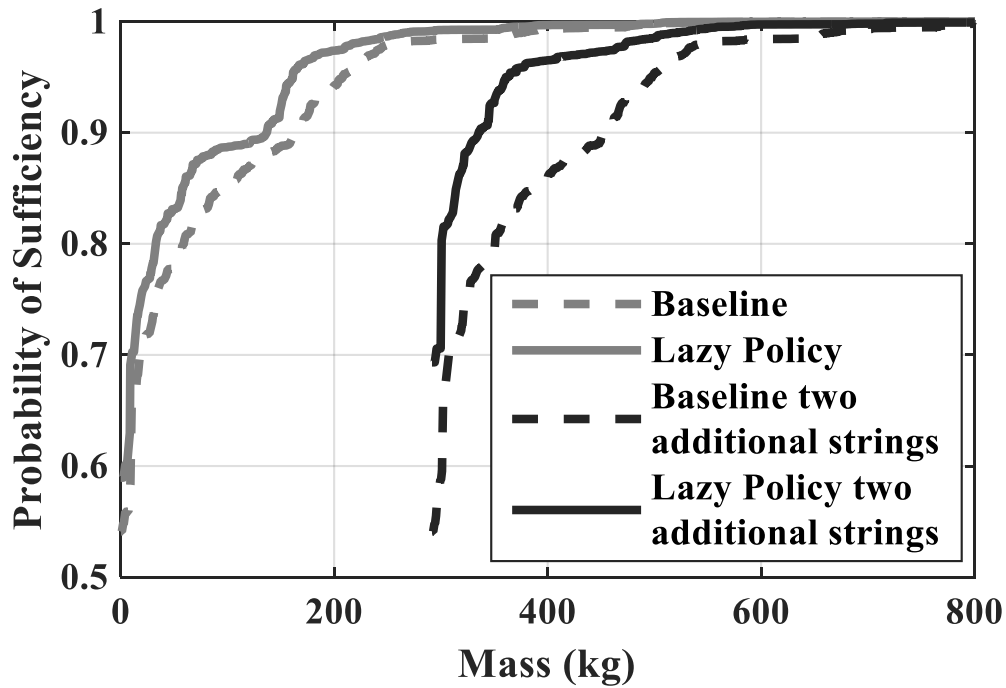


Figure 9: Additional strings results compared

Looking at the impact of sparing policy on the additional CCAA and Desiccant/Sorbent Bed strings initialization in Figure 9, the results show similar relative behavior between the baseline sparing option and the Lazy Policy with two additional strings. Again, the zero spares mass probability of sufficiency is much improved (~15% as opposed to ~5%). Like the case compared in Figure 6, Lazy Policy also improves in sufficiency at a faster rate, but the advantage is more pronounced at lower masses and mostly lost at 99.9% probability of sufficiency. These results quantify the anticipated qualitative result. Namely, when additional strings are added to a system, allowing strings to fail without immediately repairing should result in fewer spares used and increase the number of instances where no spares are required.

Another point to note is that the additional strings case is dominated at all masses by the original configuration due to the mass penalty caused by the additional strings, which indicates that including additional strings would not be an effective strategy compared to changing the sparing policies. This is because the additional strings represent spares that have been pre-deployed in a fixed configuration which limits the flexibility that these spares can be used in the event of a failure. When designing systems, the decision for such reduction of flexibility is made considering the balance between automatic failover versus human-in-the-loop diagnostics and repair. The former is likely to require more mass but put less burden on the crew and on backup systems. The latter may be more mass efficient but invites higher risk (or at least perceived risk) depending on the characteristics of the repairs that must be made (e.g., time to repair) and the backup system. Surfacing these considerations at an early stage in the design allows them to be explicitly considered and ensure that alternatives are fairly compared.

3.3.2.3 Failure Rate

Given the demonstrated performance of Lazy Policy so far, one could become predisposed to believe that this policy or some policy, can solve any performance issue. While the spare policy is a valid knob to use when addressing the challenges of developing a system to meet the challenging mission requirements, the change in sparing policy is not a cure-all. Figure 10 shows the results of applying Lazy Policy to the alternative of component failure rate increasing by 1.33 and 2 times the original amount. This case is to explore the flexibility of sparing policy in the event that the developed components do not match the originally modeled failure rate. If the components' reliability only fails to meet the originally planned failure rate by 1.33 times, the lazy policy can be used to approximate

the original baseline probability of sufficiency so that other measures are not required to mitigate the failure to meet requirements. However, in the case where the components are twice as prone to failure, switching to the lazy policy is clearly insufficient to return performance to the baseline. Overall, these examples demonstrate that relaxing the sparing expectations can reduce the number of spares required for a given situation, but it is not a cure all to overcome any underperformance.

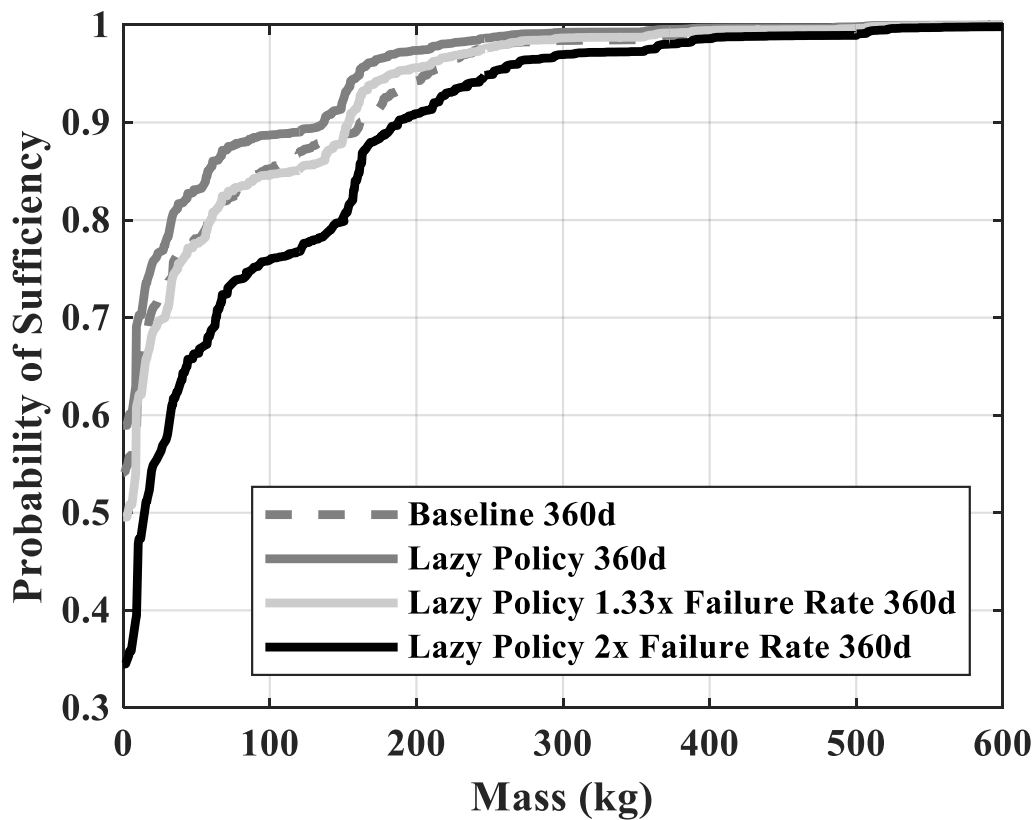


Figure 10: Lazy policy with component failure rate doubled

Having the ability to completely, or selectively, relax the strategy to spare a system or function gives decision-makers another option to respond rather than accepting an unknown risk or investing additional resources to mitigate an underperformance. The

developed approach allows the impact to be quantified and candidate responses to be tested.

3.4 Conclusion

Long duration crewed missions are likely to require significant numbers of spares to be provisioned to achieve the high probabilities of success anticipated. Given the challenges and gear ratios involved, these expectations need to be factored into early system architecting so that the success requirements can be captured, and the system impacts can be represented accordingly. For the simulation-based approach, the capability of an existing state-of-the-art tool, EMAT, has been enhanced to enable the analysis of the probability of sufficiency and spares mass in less time and thus facilitate more efficient tradespace exploration. Additionally, it improves EMAT to allow different sparing policies to be explored so that different operational risk postures and mitigations can be tested. Using this modification, a new sparing approach, named the Lazy Policy, was introduced and tested against the baseline repair-on-failure policy. The results showed that the policy can provide a decrease in the spares mass for a desired probability of sufficiency or in some cases, can be used to offset underperformance of other system parameters. Developing other general sparing policies or identifying specific policies for individual spares is left for future work, though it may be informed by policies found by the analytical method developed in the next chapter.

CHAPTER 4. AN ANALYTICAL MODEL FOR SPARING POLICY ANALYSIS AND OPTIMIZATION FOR SPACE HABITAT OPERATIONS

This chapter presents an analytical model for analyzing and optimizing spare policies as part of an overall evaluation of the probability of sufficiency for a system configuration. A key idea behind this work is to model and optimize the sparing policies using SMPs. To this end, the combination of time to initiate repair and time to repair in Semi-Markov Processes (SMPs) are represented as a random variable modeling the repair transition between two states. The shape of this random variable is dictated by the mean and standard deviation. By treating these parameters as the independent variables in an optimization problem, we can optimize how quickly a system transitions from a state with a failed component and one that is repaired while accounting for potentially adverse outcomes by delaying. In this way, the sparing policy is optimized to maximize probability of sufficiency for an allocation mass.

4.1 Analytical Methods

The analytical framework is a nested set of methods that can optimize for sparing policies. In the inner set of methods, the possible states of the system are expressed as a Semi-Markov Process (SMP) and the solution to the number of state visits is used as the basis to determine the optimal spares allocation using the modified knapsack problem. The outer set of methods searches to find the optimal set of shape parameters for the repair and buffer depletion random variables that govern transitions in the state network diagram.

Together, these methods yield optimal spare policies for a desired mass. Additionally, a method to auto-generate the state network diagram that underlies the analysis is presented.

A critical insight underlying the analytical framework is that the sparing policy is the rule set under which human intervention is initiated to repair a failed element of the system. This definition explicitly acknowledges what is often an implicit part of spares modeling; that the decision to initiate the repair of a non-functioning component or line replaceable unit (LRU) influences the total number of associated spares and consumables to meet the desired PoS. To effectively model sparing policy, how this decision process is manifested in the SMP is identified and changes to the policy are parameterized to search for preferred ones.

Core to this analysis is the modification of the parameters of repair and buffer depletion transitions, two of the three basic events in this model. Both events are modeled as lognormal distributions based on repair times and consumable buffer sizes for repair and buffer depletion actions, respectively. The third event is failure modeled with a constant hazard rate that is not modified in the analysis because it cannot be influenced by decisions of when to repair failures. The events drive the system transitions from state to state and form the basis for the formulation of the SMP. When a solution for the SMP is found, the CDF of the number of visits to each state is determined which is then correlated to the number of spares to meet a desired PoS.

In this analysis, the parameters of the basic event models are modified to change the frequency at which transitions from state to occur which impact the demand for spares and consumables. Instead of fixing the mean of the repair transition to time to repair, it is

allowed to increase and can be paired with a relaxed standard deviation to allow variability in when the repair occurs. This approach models real-world situations where the crew does not immediately react to a failure and repairs the element as other time requirements allow. The consequence of delaying repair from an SMP perspective will be existing in a state that either relies on another string performing the same function or on a backup system using consumables for longer than a specific repair action takes. In turn, such a choice may also impact the optimal number of spares or consumable load of the supporting string or backup system. By contrast, the simulation approach in the previous chapter needed the underlying logic to be modified directly to implement the specific lazy policy tested. Here the sparing policy can be tested by modifying the parameters of random variables modeling repair and buffer depletion actions of a SMP and allows the development of an analytical approach to test the sparing policy.

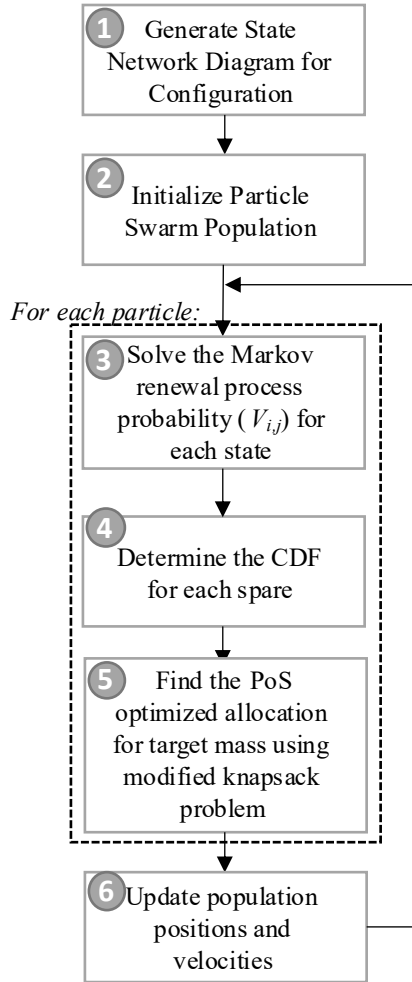


Figure 11: General analysis flow

The analysis approach centered on testing repair and buffer depletion random variable parameters is fundamentally the combination of two processes. The first is the definition and solution to an SMP for one specific set of parameter settings, represented as steps 1 and 3-5 in Figure 11. The key aspects of this part are generating the state network diagram describing the states of degradation of the system, solving the SMP to determine the likelihood of visiting each state n number times [11], convolving the states corresponding to a given spare, and for a desired allocation mass (referred to as the target mass here), determining the optimal combination to maximize the PoS [21]. This routine

is embedded in a particle swarm optimization process to vary the necessary parameter settings to test spare policy, represented as steps 2 and 6 in Figure 11. The underlying premise of this second aspect is that the traversal of the state network diagram is controlled by the parameters of the random variables governing transitions and that these parameters governing repair and buffer depletion represent the spare policy of the system. The optimization is used to search the overall tradespace to find preferred parameter settings that maximize PoS for a target mass among the set of particles. Figure 11 depicts how these two elements are merged to create the overall analysis flow.

4.1.1 Structuring the Semi-Markov Process for Sparing Policy Analysis

4.1.1.1 Generating State Network Diagrams

In the initial analysis step, a state network diagram describing the potential states of a system and the transitions that dictate changes between them is created. A state of the system in this context is the aggregate description whether each individual component of the system is currently in operation, on standby, or failed. The failure and repair rules act on the components to change their state (operating, standby, failed) which changes the overall system state. The buffer depletion state is a bit of a special case since it is a sink state that represents that an unrecoverable consequence has been reached.

The process of state network diagram generation uses states that allow the primary system to provide the desired function, called valid states, as an input and seeks to describe when the system is and is not in a valid state. While this step can be accomplished by hand, a computer-led approach was implemented to improve traceability, reduce human error, and scale to large (~1,000 state) state network diagrams. To enable the computer to perform

this step, transition rules are formalized to describe the conditions under which a given event can result in a transition from the current state. Other work describes these as event policies [22], but this terminology is avoided to prevent confusion with the concept under study, sparing policy. A set of transition rules describing failure, repair, and buffer depletion transition for elements within a candidate state and the state itself. A representative set of transition rules is shown in Table 5.

Table 5: Transition Rules

| Transition Action | Is the current state a valid operational state? | Is the element within the current state functioning and operating? | Change to New State | Options for new state |
|--------------------------|--|---|--------------------------------------|---|
| Failure | Yes | Yes | Element is now failed | Select a new valid state not using a failed element or the same state just with the element failed |
| Repair | N/A | No | Element is now functional | Select a new valid state with repaired element operational or the same state just with the element repaired |
| Buffer Depletion | No | N/A | Depletion sink state for that buffer | New state only if the first instance of this buffer depleting |

The transition rule implemented for failure is that element n can fail if the current state is a valid operational state and element n is in operation. If so, the transition is triggered, and element n is marked as failed. For the element repair basic event, the transition is triggered if the element is already failed to result in a new state where the element is either on or off depending on the overall configuration of the system. Finally, buffer depletion can occur if the current state is not a valid state meaning that the backup utilizing the consumable buffer is in use and the available amount is exhausted. When this transition is triggered, the transition leads to a common depletion state for all transitions associated with this depletion type. Note that for each transition rule, the type and parameters governing the transition triggered by the rule is *not* considered. The rules are capturing whether a transition from an existing state to a new state can occur, not the likelihood or frequency of the discovered transition. Later, when the parameter space for each transition is defined, it will be possible to allow parameter settings for a transition that preclude it from being triggered in the time horizon of the specific problem. This dichotomy allows the state network diagram discovery process to occur once and ensures that the same underlying state space is used for each particle and generation of the policy optimizer. With the transition rules defined, state space exploration can begin in earnest.

One other innovation of this work is the introduction of ghost states to the state network diagram. These are transitory states entered because of a repair action whose destination is another state that can be entered via multiple destinations, like renewal to the initial state. These ghost states are immediately exited via a Kronecker delta function. These states are inserted to assist in the counting of repair actions, the importance of which will be discussed later.

To build out the SMP model, a valid starting state from the list of all valid states is added to the state queue and that queue is processed in a first-in-first-out (FIFO) manner which drives a breadth-first generation of the state network diagram. A test state is taken from the front of the queue and then each transition rule for each action is compared against it to see if a transition is triggered, and if so, the resultant state is added to the state queue and the state network diagram is updated. By examining all possible basic events element by element, the resulting search is breadth-first. The overall process is explained as part of the pseudocode below utilizing the transition rules in Table 5. The pseudocode also utilizes a depth parameter to limit the size of the state network diagram. Essentially, this parameter stops adding failures to the state queue once a certain depth in the state network diagram.

State Network Generation Pseudo Code

Input: Valid Configurations, Transition Rules, Initial State

Parameters: Depth Parameter

Initialize: State Queue with Initial State as first entry; State Network Diagram with Initial State as first entry

Repeat

Step 1 Remove first state in State Queue; this state becomes the Test State

For Each Component

Step 2 Apply Failure Rule to component in Test State

Step 2.1 If rule triggered and depth in State Network Diagram \leq Depth Parameter; then, Add a new state with component failed to State Queue and State Network Diagram

Step 2.2 Else, rule not triggered or depth $>$ Depth Parameter; then, do nothing

Step 3 Apply Repair Rule to component in Test State

Step 3.1 If rule triggered and new state \neq initial state; then, Add a new state with component repaired to State Queue and State Network Diagram

Step 3.2 Else if, rule triggered and new state = initial state; then, Add new entries to State Network Diagrams connecting test state to initial state with ghost state in the middle

Step 3.3 Else, rule not triggered; then, do nothing

End

For Entire Test State

Step 4 Apply Buffer Depletion Rule to Test State

Step 4.1 If rule triggered; then, Add entry to State Network Diagram to connect Test State and Buffer Depletion State

Step 4.2 Else, rule not triggered; then, do nothing

End

Until State Queue Empty

Step 5 Post process State Network Diagram to enumerate the transitions between the states found in Steps 2-4; capture as Transition Matrix

Step 6 Post process State Network Diagram to capture the implied graph as the Adjacency Matrix

Output: State Network Diagram, Transition Matrix, Adjacency Matrix

The Depth Parameter is used because the process of finding new states which are then added to state queue can potentially continue in perpetuity if there is no mechanism to stop adding new states to the queue. Owens and de Weck [15] addressed this by solving the SMP for the in-process state network diagram each time a new state as added to the state network diagram and pruning any states that were not visited with a minimum defined likelihood. This approach needs fixed parameters for the transition random variables, is computationally expensive as the SMP must be solved each time a state is added and has the potential to prune states that may be visited under a different spare policy. To mitigate

these drawbacks, a depth-based rule is used that stopped adding failed states to the state queue once a certain level of the state network diagram tree is reached. Once that depth is reached, only repairs or buffer depletions can occur; the former renewing back to the initial state and the latter leading to depletion states that act as sinks. In both cases, new state generation rapidly diminishes, and the state queue is emptied. This results in the state network diagram used in the analysis, which can be used to construct the transition and adjacency matrices, and both are exported. A brief example of the state network diagram generation process is presented in the subsequent section.

For the analysis of the case study, the depth parameter was set to two which means the primary system can have up to two failures in it before repairs must be made to restore the system to its initial state with all components in a functional state. This setting results in 1075 states and 1887 entries in the state network diagram. Setting the depth parameter any larger would have been computationally unmanageable for the system used in the analytical evaluation, a Mac Studio with an Apple M1 Max chipset and 32GB of RAM running MATLAB 2022a in the Rosetta 2 translation environment.

The three matrices outputted from the state network diagram define the degradation and repair of the system based on the valid states the system can operate in and the transition rules that described how elements can fail, be repaired, or be depleted. The state network diagram forms the foundation of the overall SMP analysis. The states in the network diagram are the different configurations the system can enter based on the actions of failure, repair, and buffer depletion described by the transition rules in Table 5. The rate at which transitions occur are described by random variables whose parameters influence how often a given state is visited. In the analysis, the parameters of repair and buffer

depletion transition random variables represent the repair policy as they are what model the crew behavior towards repairing components and the ability of the backup system to support the repair approach. Since the transitions are the core to what is being tested, the state network diagram is held constant across the spare policy optimization to ensure the parameters being tested are driving the change in behavior. These parameters are randomly initialized as members of a particle swarm, discussed further in Section 4.2, and for each set of parameters, the state network diagram is evaluated to determine the PoS.

4.1.1.2 State Network Diagram Generation Example

Take as an example the simple system in Figure 12 below and assume there is a backup system requiring a buffer. Components A and B are in serial, so both must be operating for the system to be operating. That means the only valid state for this system is 11 (column 1 represents component A; column 2 represents component B) and would be used as the initial state in the state queue for the state network diagram generation process. For this process, we will use a depth parameter of 2.

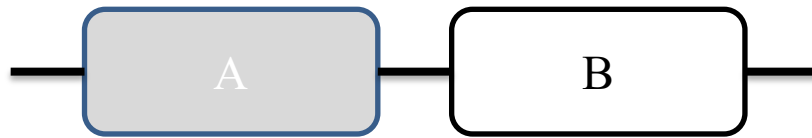


Figure 12: System Configuration

Applying the rules in Table 1 (see main paper) to the initial state will start to build out the state network diagram. Initially, the rules for component A are applied to the first column of the state since that is the representative for component A. First the failure rule is applied to and is triggered because 11 is a valid state and A is operating (denoted by the

1 in the first column). A new state, -11, is added to the state queue because the current depth is 0. Next, the repair rule is applied but a new state is not found because A is operating is the current test state, 11. Now we repeat the process for component B and similar results are found. As a result, another state, 1-1, is added to the state queue. The state is valid, so the buffer depletion rule is not triggered when applied. As a result, the state network diagram currently looks like Figure 13 below.

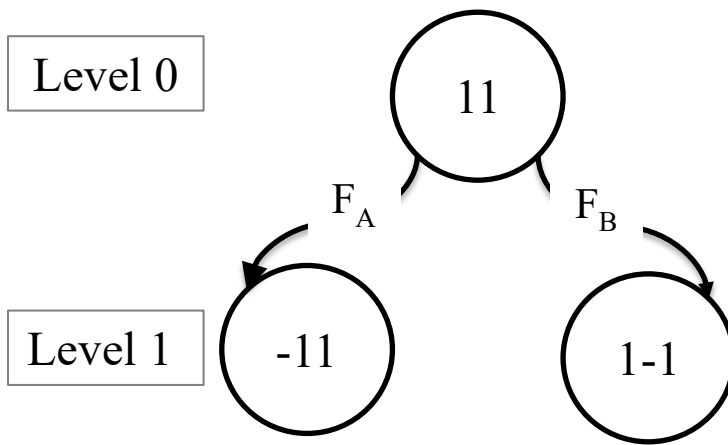


Figure 13: State Network Diagram after 11

The state, -11, at the top of the state queue becomes the next test state. The process begins again with column 1 representing component A. It is currently failed, so it cannot fail again. When the repair rule is applied, a repair transition is found. Since the resulting state from repair, 11, is the same as the initial state a new state is created in the state network diagram but not in the state queue. Ghost State 1, GS1, is entered because of the repair of A and is exited immediately using a Kronecker delta function. This state is used to count the repair action of A since SMP does not have a natural method for counting transition occurrences but is able to count state visits. With the repair rule handled for component A, the failure and repair rules are applied to component B of the test state. Neither rule is

triggered because the test state is not valid and component B is not failed. However, the buffer depletion rule is triggered because the test state is not valid meaning the backup system with the buffer must be utilized and its buffer could deplete. The state network diagram after this test state is in Figure 14.

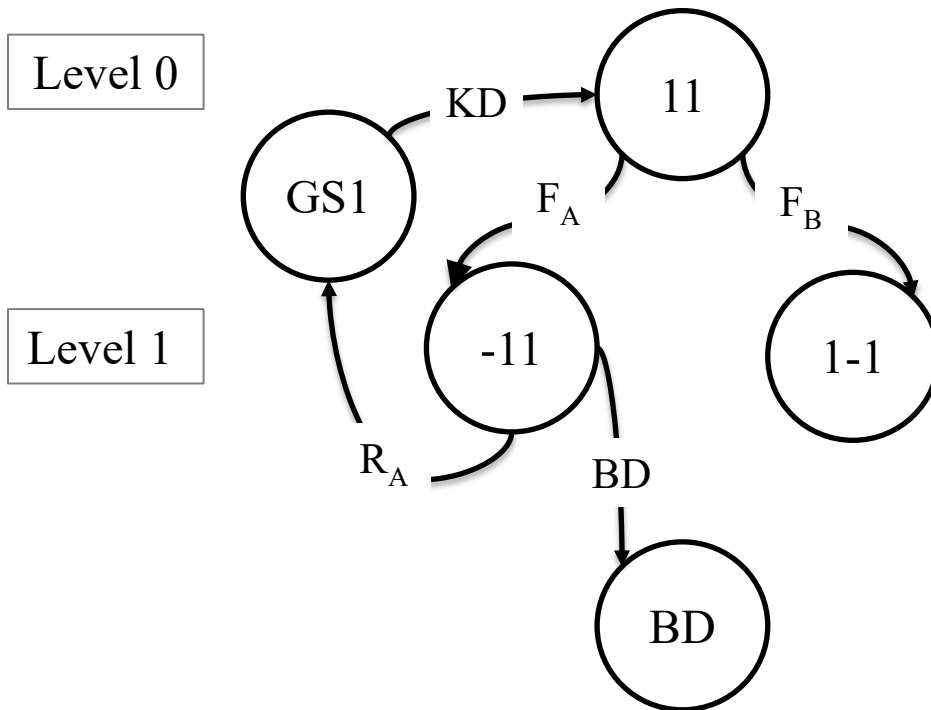


Figure 14: State Network Diagram after -11

The next state in the state queue is 1-1 so it becomes the test state. The failure and repair rules for component A are not triggered when applied because the state is not valid and A is not failed, respectively. For component B, it is already failed so it cannot fail again. For the repair rule and component B, repair is triggered and the resulting state, 11, is again the initial state so Ghost State 2 is created as the intermediary between the test state and the initial state. The buffer depletion rule is also triggered so a transition is added

between the test state and buffer depletion state. Now that all rules have been applied to the test state, the resulting state network diagram is shown in Figure 15: State Network Diagram after 1-1

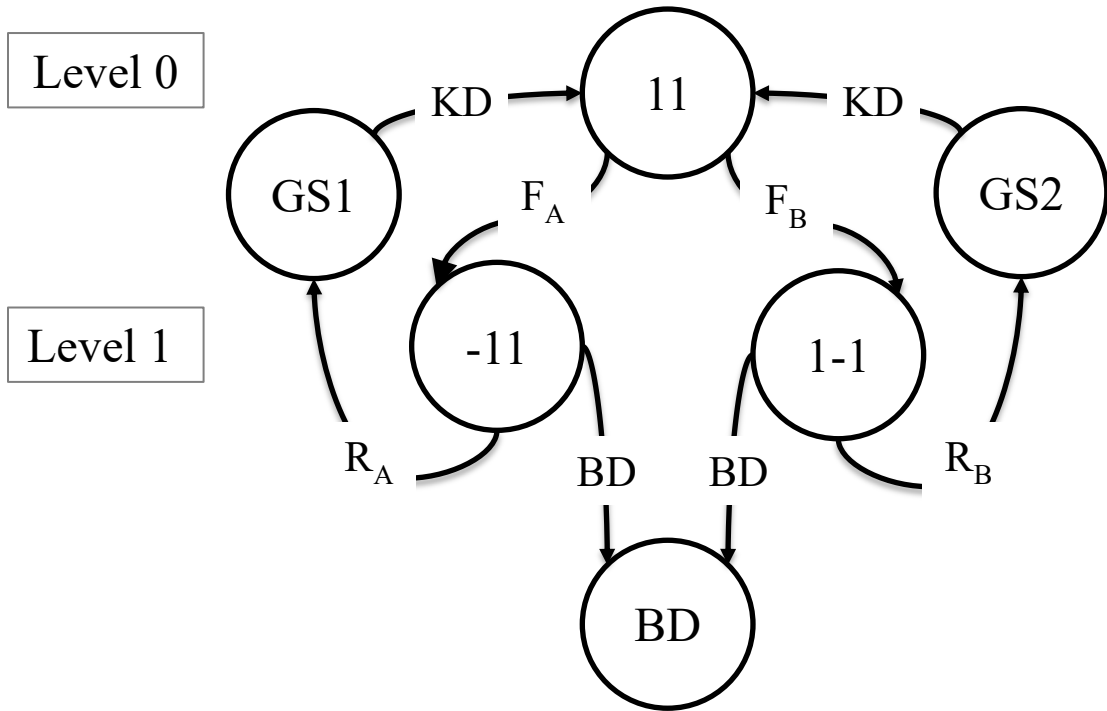


Figure 15: State Network Diagram after 1-1

Since there are no other states in the state queue, the state network diagram generation process is complete.

4.1.1.3 Evaluating the State Network Diagram

Given the generated state network diagram by the process in Section 4.1.1.1, there are a set of actions that solve the SMP for the number of visits to each state in the state network diagram and then allocates the visits associated to repair to evaluate the PoS. The

first step in this process is to determine the Markov Renewal Process Probability, $V_j(\mathbf{k}, t)$, which describes the likelihood of visiting state j , k times, over time t . Owens [15] leveraged Warr and Collins' [14] overall solution approach which leverages Abate and Whitt's numerical Laplace solution approach [23] and added numerical convolution to determine the difference distributions. These difference distributions are used to describe the random variables for transitions that cannot rely on an assumption of memorylessness. These random variables and their corresponding cumulative distribution functions are assembled into two matrices, \mathbf{f} and \mathbf{F} , which capture the distribution of each transition. These matrices are used to determine the kernel matrix, $\mathbf{Q}(t)$, which captures the joint probability of the transition occurring given the other transitions that could occur from that state starting at $t=0$. This matrix is assembled using Equation 4 based on the assumption that transitions can be modeled as competing continuous transitions where the earliest occurring transition is selected. In SMP literature [24], this is called a type II transition and contrasts with a type I transition where discrete transition probabilities are compared and once one is selected the time to transition is then determined using the PDF for that transition. Type I is more appropriate for problems where only one path can be selected at a time with the selection and time to transition being two distinct random variables while type II is more appropriate for reliability problems where the time to transition is described by competing, simultaneous processes modeled with random variables.

With the kernel matrix created, the unconditional waiting time density matrix, $\mathbf{H}(t)$, can be assembled using Equation 5. Each entry in H describes the time spent in state i unconditioned at the destination state or states that can be entered from i . This probability distribution function assumes that i was most recently entered at $t=0$.

$$Q_{i,j}(t) = f_{i,j}(t) \prod_{k \neq j} (1 - F_{i,k}(t)) \quad (4)$$

$$H_{i,i}(t) = \sum_j Q_{i,j}(t) \quad (5)$$

To determine the Markov Renewal Process Probability, the \mathbf{Q} and \mathbf{H} matrices must be transformed into the Laplace domain. When these matrices, and others, are in the Laplace as part of calculation, it is denoted by a tilde, \sim , above the relevant term. Warr and Collins detail the specific numerical approach to take the Laplace and inverse Laplace of various terms. With \mathbf{Q} in the Laplace domain, the Laplacian PDF of the first passage time matrix, $\tilde{\mathbf{g}}(s)$, is calculated using Equation 6 which is then used to determine the Laplace of \mathbf{V} shown in Equation 7. In Equations 6 and 7, \mathbf{I} is the identity matrix, $\mathbf{1}$ is a matrix of ones, and \circ is an operator for element-wise multiplication, also called the Hadamard product.

$$\tilde{\mathbf{g}}(s) = \tilde{\mathbf{Q}}(s)(\mathbf{I} - \tilde{\mathbf{Q}}(s))^{-1} \left[\mathbf{1} \circ (\mathbf{I} - \tilde{\mathbf{Q}}(s))^{-1} \right]^{-1} \quad (6)$$

$$\tilde{\mathbf{V}}(k, s) = \frac{1}{s} \left(1 - \tilde{\mathbf{g}}(s) \circ \left[\mathbf{1} (\mathbf{I} \circ \tilde{\mathbf{g}}(s))^k \right] \right) \quad (7)$$

The Markov Renewal Process Probability is found by completing the numerical inverse Laplace Transform of $\tilde{\mathbf{V}}(k, s)$ and yields a cumulative distribution function (CDF) for each state. Once in the time domain, k is the number of times a given state j is reached starting from state i in time t , which is typically selected as the mission duration. The initial

state is the operating state at the start of the mission and is also used as the first state in the generation of the SMP diagram.

With the CDF of visits for each individual state, the next step is to determine the CDF for each spare. The need for a certain spare will correlate to visiting a given set of states. Since V_j has already been calculated for all states, this is just a matter of convolving the CDFs for all states corresponding to a given spare. Convolution is multiplication in the frequency domain, so the standard MATLAB Fast Fourier Transform (FFT) and inverse Fast Fourier Transform (IFFT) functions are used to perform this calculation. This approach showed the improved speed when compared to the convolution function which implements a linear convolution calculation versus the circular convolution of FFT/IFFT.

The selection of which states correspond to a given spare will impact the results. This limitation is driven using the SMP formulation and the solution therein which only allows for metrics to be calculated for state visits, not transition triggers. In Owens' work [11], he selects all states entered via failure transition corresponding to components that can be replaced by the given spare. This approach is appropriate for a spare-on-failure policy; however, because the purpose of this work is to test policies and potentially never repair certain failures, correlating failed states to spare needs would not be appropriate. Instead, the correlation is on states entered by repair. Specifically, for states only entered by one transition, the state can be counted directly, whereas for states entered by multiple repairs, the renewal to the initial state, for instance, ghost states are inserted between two states connected by repair transitions in the SMP generation step. The ghost state is entered via the repair transition and is then immediately left using a transition modeled with the Kronecker delta function. As a result, the ghost states directly map to the repair action for

a corresponding spare which can then be grouped and convolved as described above. This results in a more direct measure of spare needs with a small state growth penalty. For mission durations approaching infinity and at very high probabilities of sufficiency, the two approaches to selecting the states will converge but the ghost state approach was selected given the interest in sparing policy.

With the CDFs, S , for the individual spares, s , demand determined, the best PoS for the target mass can be determined. Recall that PoS is the likelihood that the allocated spares are adequate to meet the spares demands of the mission as dictated by the unreliability of the system and the sparing policy employed. As in the previous chapter, the modified knapsack problem is used to determine the allocation that maximized PoS over a range of masses and denote this context by using $PoS_{MKP,A}$. In short, Equation 8 is determined by using the subproblem shown in Equation 9 over one kg mass steps, d , from 0 to the capacity c . In the simulation implementation, the PoS of a given allocation X was determined by how many simulated missions would have had sufficient spares if started with the test allocation. For the analytical approach, simulation results are not used in the calculation of PoS. Instead, the number of spares, x , for each spare type s is used as an input to the corresponding CDF and the product of those results give the overall PoS for the allocation as shown in Equation 10.

$$\text{maximize } PoS_{MKP,A}(X)$$

$$s. t. \sum_{s=1}^n x_s m_s \leq c \quad (8)$$

$$x_s \geq 0$$

$$PoS_{MKP,A}(X_{d,s}) = \begin{cases} PoS_{MKP,A}(X_{d,s-1}) & \text{if } d < m_s \\ \max_{b=1, \dots, \lfloor \frac{d}{m_s} \rfloor} \{PoS_{MKP,A}(X_{d,s-1}), PoS_A(X_{\lfloor (d-l \cdot m_s), s-1} + b \cdot x_s)\} & \text{if } d \geq m_s \end{cases} \quad (9)$$

$$PoS_A(X) = \prod_{\forall s} S_s(x_s) \quad (10)$$

In this problem, c is determined by the target mass minus the derived mass for the first instance of consumables that are defined by the buffer depletion parameters. This decrement ensures that the buffer represented in the state network diagram starts full for at least one pass through the state network diagram. Subsequent passes are captured by including the buffer in the optimization described by Equations 8 to 10. When working with one buffer, parameter maximums can be used to ensure the buffer mass does not exceed the target mass. This approach does not work for problems with multiple buffers as the sum of the masses can exceed the target. In those problems, a penalty function is used, and when the buffer mass exceeds the target mass, the modified knapsack problem is not solved, and PoS is set to -1 and infeasibility is returned. Otherwise, the optimal allocation to maximize PoS for the available mass is determined as described in Section 3.1.2.

4.2 Optimizing Spare Policy

The evaluation described in Section 4.1.1.2 is the process for determining the PoS for one set of repair and buffer depletion parameters representing the sparing policy (e.g.,

how much the crew “delays” the repair after a failure). Specifically, the μ and σ parameters defining the lognormal random variables modeling repair and buffer depletion transitions are varied to find the optimal PoS at a target mass. This contrasts with previous work [11], where the repair parameters were fixed to capture the mean time to repair with a small deviation, which modeled a repair on failure policy.

The policy optimization problem is described in Equation 11. The optimization finds the best PoS for the desired mission duration, t_T , and mass target, m_t , within the feasible parameter space for a specific target mass by varying the parameters, ζ_p , which are collected in the vector \vec{Z} . These parameters are associated with the repair and buffer depletion transitions of the kernel matrix, \mathbf{Q} , as defined in Equation 4. Among the state transitions f in the definition of \mathbf{Q} , the parameters for the repair and buffer processes are optimized and represented by $f_n = \text{lognormal}(\zeta_{2n-1}, \zeta_{2n})$ using the varied parameters in \vec{Z} , while those for the failure processes remained fixed. Additionally, the sum of the masses of the consumables as a function of the associated buffer depletion parameters must be less than the target mass since the first set of consumables must be preallocated as discussed before. For each particle at each step, the SMP is repeated for the new \vec{Z} which results in a new $PoS_{MKP,A}$ for the target mass which becomes the PoS_{SMP} in Equation 11.

$$\text{minimize } 1 - PoS_{SMP}(t_T, m_t, \mathbf{Q}(t, \vec{Z}))$$

subject to

$$\zeta_{p,min} \leq \zeta_p \leq \zeta_{p,max} \tag{11}$$

$$\sum_{n \in \text{consumable type}} m(\zeta_{2n-1}, \zeta_{2n}) \leq m_t$$

A particle swarm optimization (PSO) approach was selected since little is known about the shape of the underlying problem which favored using an approach capable of searching the overall solution space. The overall implementation does not deviate significantly from the standard PSO approach but there were some specific choices to tailor to the spare policy problem.

PSO is used to search for preferred parameter settings of repair and buffer depletion random variables which are the representation of sparing policy. Originally proposed by Kennedy and Eberhart [25], PSO mimics the behavior of insects in a swarm where each individual, called a particle, performs its own search of the tradespace under the influence of its own history as well as that of other particles. This behavior is accomplished by encoding the parameters as a position of the particle and giving the particles a velocity that changes based on the current velocity, the position of the particle's best-found location, and the position of the current global best among all particles. Like a swarm, the particles will converge on the intended target, an optimum within the tradespace. In this case, the goal is to maximize the PoS for a fixed allocation mass by varying the approach to sparing represented by the repair and buffer depletion parameters encoded in the particles.

4.2.1 Initializing the Population

The PSO is initialized with particle positions, x , and velocities, v , to start searching for an optimal PoS for the target mass. The particles are initialized using Equations 12 and 13.

$$x_0^l = x_{min} + rand(0,1)(x_{max} - x_{min}) \quad (12)$$

$$v_0^l = \frac{x_{min} + rand(0,1)(x_{max} - x_{min})}{\Delta t} \quad (13)$$

Both repair and buffer depletion actions have well-defined minimums. The lower bound for repair actions is the time to repair of the spare and for the buffer depletion, there must be at least one canister for the state network diagram to be valid. These constraints can be converted into the corresponding μ and σ for the lognormal distribution and used as the minimums. For the buffer depletion parameters, the maximums are also known and used in the bound handling step as well as initialization. For repair transition bounds, the maximum is estimated to push μ beyond mission duration and σ is set at an arbitrary amount, natural logarithm of 5, to approximate a random variable with a large standard deviation relative to the mean.

A different approach is used for the buffer depletion steps whose limits from a mass perspective are set between 0 kg and the target mass. How these values are manifested in the lognormal random variables depends on the consumable being modeled. Owens developed a model for tanks based on the processing or consumption rate for the substance in the tank and a small standard deviation. A similar model is used for discrete consumable elements like LiOH canisters that adsorb CO₂ in that the consumption rate is used to define the number of canisters available in each pass through the state network diagram. Unlike tanks that can be refilled, these canisters cannot be refilled, an assumption made in previous analyses, instead the number of canisters defined by the buffer depletion parameters is treated as a set that can only be used during that pass through the state network diagram.

This approach is conservative because the consumable may not be spoilable over that timeframe, but the optimization approach should limit excessive canister allocation.

With the particles defined, the PoS for each member of the swarm is found using the process described in Section 4.1.1.2. This set of values is used to initiate the PSO process in earnest.

4.2.2 Particle Swarm Optimization

In each iteration of PSO, the PoS results are compared to both the best result found for that particle and for all particles and if the new result is better, it becomes the new best. To find the optimum associated with Equation 11, the best is being observed for both an individual particle and for the population as a hold at each time step. If a new best is found, the “flight path” of the particle is subtly altered otherwise the particles start to do a more localized search near the best found to date. This behavior is driven by particles’ positions and velocities updates based on the algorithm and bound handling procedures. Equations 14 and 15 describe this process where the new velocity for the particle is determined, which is then used to update the position of the particle, which defines the parameters being tested. The velocity is influenced by the location of the particle relative to the current local best, p^l , and relative to the current global best, p^g .

$$v_{k+1}^l = wv_k^l + c_1rand(0,1)\frac{(p^l-x_k^l)}{\Delta t} + c_2rand(0,1)\frac{(p_k^g-x_k^l)}{\Delta t} \quad (14)$$

$$x_{k+1}^l = x_k^l + v_{k+1}^l \cdot \Delta t \quad (15)$$

The constants w , c_1 , and c_2 in Equation 14 are set at 0.5, 1.5, and 1.5, respectively, based on the recommendation of previous work [26]. Additionally, Δt is set to 1 as is typical but included for completeness.

Occasionally the velocity of the particle can result in a new position that exceeds the limits on individual x values, which particle swarm treats as a bound handling problem, not as a true constraint violation. If a particle exceeds or is going to exceed a positional bound it can be corrected in several ways. The method selected for this work is the reflect bound handling approach referred to as Reflect. In Reflect, the particle dimensions that exceed the bounds are reflected over the respective bound back into the feasible space. The corresponding velocity of the reflected dimensions is set to zero to stifle the motion that led the particle to leave the feasible space in the first place. There are other bound handling techniques that could be employed, but experimental work by Helwig et al. [27] suggested that Reflect is appropriate as a generic approach for this problem. If more was known about the underlying problem, one specific approach could be chosen.

A different approach must be used for the constraint on total buffer mass, the last constraint in Equation 11. As discussed above, the mass of each consumable buffer is a function of the parameters in the step and then resulted masses are summed to see if they exceed the target mass. A simple penalty function is implemented where PoS for the particle is set to a value of -1 if this constraint is violated. This has the effect of yielding an unnaturally high value of the function to be minimized which combined the previous best terms in the velocity update equation in Equation 14, resulting in a rapid return to the feasible space. This method plus bound handling for the other constraints ensures that the policy found is within the feasible space.

In summary, the analytical approach implements a standard PSO algorithm with problem-specific constraints. Rather than letting particles reach positions that are infeasible and may not have calculable PoSs, a bound handling approach is implemented.

4.3 Results

To baseline the performance of the analytical approach, two cases are tested, corresponding to the baseline (repair-on-failure) and lazy policies identified in the previous chapter, without the PSO. In this work, the parameter settings for repair and buffer depletion were used to represent the baseline and lazy policies. To represent the baseline policy, which initiates a repair upon failure of that component, the parameters are set to the minimum time to repair with low variance and a correspondingly low number of backup consumables (LiOH canisters). Similarly, to represent the lazy policy, which waits until the backup system was being utilized and only repairs enough to return the primary system to an operational state, the parameters corresponding to the mean time to repair of the components to the primary string of the common cabin air assembly (CCAA) were set to a value longer than the mission duration for repair transitions between the first failure state of the component and the initial state only. Since the CCAA is the only redundant string within the CDRA, these components of the primary string are the only ones that can fail and lead to a state that does not rely on the backup system. In those states, the repair is not desired when utilizing the lazy policy, so the parameters are adjusted accordingly. With those parameter settings established, the analytical approach described in Section 4.1.1.2 is used and the results compared to the simulation results from the previous chapter are shown in Figure 16.

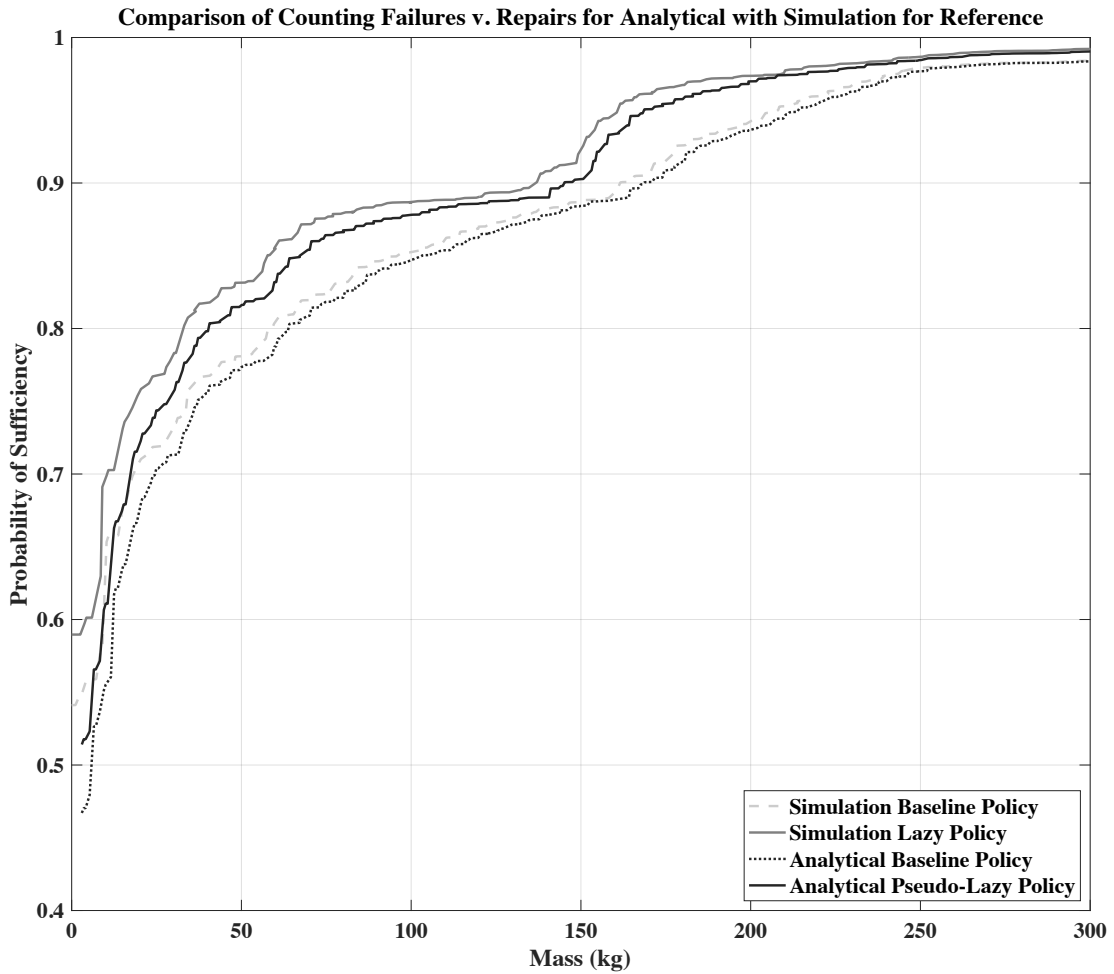


Figure 16: Comparison of analytical and simulation results

As Figure 16 shows, the analytical results are similar to the simulation policies they are attempting to replicate. There are small differences between these cases, which can be explained as follows. One of the main drivers of difference, present for both the baseline and lazy policies, is related to canister methodology. Recall that for the analytical approach, the number of canisters is determined by the buffer depletion parameters and all the canisters must be consumed on a conceptual pass through the state network diagram. In practice, this means that canisters must be added in sets rather than one at a time. This causes a major divergence at very low masses where the need to pre-provision the first set

of canisters to ensure the validity of the state network diagram does not allow canister-by-canister optimization available in the simulation. Once the mass exceeds 25kg, the performance of the analytical approach is much closer, but this limitation still drives some differences. Additionally for the lazy policy, the choice to use a depth parameter of two results in the original version of the lazy policy not being perfectly replicated. Recall that for the lazy policy repairs are only made to make the primary system functional again. But for the analytical approach with a depth parameter of two, the structure of the state network diagram requires renewal to the initial state and cannot pause in a partially repaired state once repair states are reached. Delaying the initiation of repairs, as discussed above in the setting of parameters, can replicate most of the lazy policy but the limitation on the states drives a larger difference versus the baseline policy comparison at lower masses. This behavior could be mitigated by increasing the depth parameter, but the number of states generated was too large for the compute resources in this case.

4.3.1.1 Performance of Proposed PSO Sparing Policy vs. Conventional Sparing Policies

With the comparisons to previous work established, the PSO is used to find a better policy than either the lazy or baseline policies. Where the lazy policy has already been identified as superior to the repair-on-failure at low masses, the parameters for the lazy policy are included in the initial particle swarm population with all other members randomly initiated in the feasible space. For higher masses where the baseline policy is better due to the limitations discussed above, the parameters for the baseline policy are seeded into the initial population. Additionally, a target mass for optimization must be selected and the corresponding results were sensitive to this choice. Looking at Figure 16, at low masses there is a wide range of results that can be achieved based on the policy

selected. At higher masses, the impact is lower as spares are sufficiently provisioned to meet either policy. To test the impact of the results, three target masses are optimized: 100, 400, and 700kg. The results and the comparison to the non-optimized analytical results are shown below in Table 6.

Table 6: Probabilities of Sufficiency for Different Sparing Policies

| Target Mass | 100kg | 400kg | 700kg |
|-------------|-------|-------|--------|
| Baseline | 84.6% | 99.3% | 99.98% |
| Lazy | 87.8% | 99.5% | 99.89% |
| Optimized | 88.0% | 99.6% | 99.98% |

At the lower allocated masses, the optimized policy corresponding to that target performs better than not only the baseline policy but also the lazy policy, which supports the hypothesis that there exist better policies beyond the conventional baseline and even the recently-proposed lazy policies. For the target mass of 700kg, the baseline policy, repair on failure, is confirmed as the optimal policy. This result suggests that when very high spares allocation masses are available that the baseline policy can be optimal. Note that the optimized policy is determined for each target mass; when the optimized policy for a particular target mass is applied to another target mass, then this policy would not be optimal. In other words, the \vec{Z} found for one target mass is not equal to the \vec{Z} found for another target mass. This finding further cautions against generalizing a given policy as

best across all mass allocations. Rather, the overall mass available will influence the repair policy utilized, or conversely, the desired PoS and associated probability of mission success will dictate both the number of spares allocated and the repair policy used to maximize the efficiency of those spares.

The main driver for improvement of the lower target mass cases is repair of some components late in the mission duration (<60days remaining) that would never be repaired in the lazy case. Essentially, the optimizer found a crossover point in time where it is prudent to repair certain components of a redundant string rather than letting it continue to remain unrepaired. This finding is not a particularly concise heuristic that can be summarized as a part of a named strategy, like the lazy strategy, because it is dependent on the failure rate of certain components and the commonality across the system of the spare used. This complexity highlights the tradeoff between the optimality of the sparing policy and its intuitiveness; the optimized policies achieve a better performance at the cost of being less intuitive or straightforward to implement than the baseline or lazy policies.

Lastly, it is worth noting that although the proposed sparing policy optimization method was motivated by the early-stage system design challenge, it can also be used as an operational mitigation when design targets, such as failure rate, are not met and additional resources (i.e., development dollars, additional spares mass) are not available to mitigate as demonstrated in our previous work [3]. As more is learned about the system, this process can be repeated to identify the best approach to maximize probability of sufficiency as other areas become less and less flexible.

4.4 Conclusion

Crewed missions with limited opportunities to resupply or abort will need an integrated sustainment approach to achieve the high probabilities of mission success desired. With the mass and cost penalties for each kilogram of mass added, the spares mass necessary to meet the desired probability of success, or PoS as a proxy, should be included early in the design of the system to ensure that the design meets the expected requirements and that trade studies are able to have fair comparisons.

Building upon the findings of the simulation-based approach that the repair policy used can influence the mass required to achieve a certain PoS, an analytical approach is created to test sparing policies parametrically and is overlaid it with a PSO to determine preferred sparing policies for targeted masses. The analysis revealed that the policy preferred shifts with the target mass and that should be accounted for when identifying policies for implementation.

CHAPTER 5. CONCLUSION

A premium is placed on ensuring that there is a high likelihood of mission success for crewed missions. As we endeavor to complete missions that further push the limits of available technology, the conflict will grow between that desire and ability to provision the resources to prevent small failures from accumulating into overall failure. This tension increases the need to not only include estimates of spare allocations to achieve the desired reliability but also codify the sparing policy used to maintain the system in operation.

The totality of this research provides a foundation for doing this for future architectural and system trades. By utilizing real system configurations, modelers can leverage knowledge of existing solutions, the likely point of departure for follow-on systems, and capture the true dynamics of the multi-tier redundancy often present in space systems. While this complicates the initial data collection and models somewhat, it allows for the exploration of sparing policy which has a demonstrable impact of the results generated. Further, by providing both simulation and analytically based solutions, optionality is provided to the evaluator to fit the approach to the specific problem at hand.

For the simulation-based approach, a number of improvements were made to the existing EMAT approach. Firstly, the MKP optimization approach replaced the greedy algorithm which allowed probability of sufficiency and the associated spares allocation to be found at all masses. As part of this shift, the calculation of the objective function also shifted from one based on the components driving failure to one based on the allocations allowing success to occur. This improvement allowed a converged set of simulation results to determine the entire Pareto front between mass and PoS versus the stepwise approach

required by the Greedy algorithm which required a new set of simulation results at each mass step. This updated approach was paired with an improvement to the discrete event simulation activity flow that enabled new spare policies to be tested. Specifically, the addition of a repair queue allowed for the concept of waiting until a predetermined time or condition was met to initiate a repair action.

With the improved simulation implemented, a new sparing policy was tested against the baseline policy of repair upon failure. This comparison was against several scenarios where parameters such as days to repair, failure rate, and system configuration were varied. The results demonstrated that sparing policy can have an impact as large as these other parameters and can potentially serve as an operational mitigation if planned capabilities in other areas are not met.

This change gave substantially more control within the simulation. The positive aspect of this is that the rules governing a transition can be specified down to the component level and can consider the state of the system and the other components in the queue. This makes the simulation-based approach ideal in situations where the policy is relatively well-known, and the modeler may want to make small changes either to policy or system configuration. Additionally, the similarity between the discrete event simulation and the actions taken in real-life along with the structure of sparing policy rules in the tool makes explanation to non-subject matter experts straightforward. This legibility makes it easier to get buy-in into the results of the simulation and makes actual implementation of the policy easier in practice. This is especially important when discussing results with crew whose lives may depend on the decisions assisted by the analysis.

However, the flexibility and legibility this approach affords comes with challenges. The rules must be conceived of by the modeler, and then the code of the simulation must be updated rule by rule to implement the policy. With the changes in place, the simulation can be run hundreds of thousands of times over. If one wants to test different policy, this process must be completed each time to make the comparison.

The analytical approach described in this work addresses the need to test different policies against one another. This was accomplished by constructing the problem as a Semi-Markov process and recognizing that the sparing policy was encoded as the parameters of the distributions describing the likelihood of performing a repair action or experiencing buffer depletion. Additionally, this research describes a process for automatically generating the state space and associated matrices underlying the SMP problem, which allows for more complex state spaces to be discovered and explored while reducing the possibility of human error in the process of creation. With the problem described, the solution to the SMP is found and those results could be as an input to the same MKP optimization process with a problem specific objective function. This provided the optimal spares allocation for the specific random variable parameters used without having to run any simulations. This process was then wrapped with a particle swarm optimization routine which varied the parameters of the repair and buffer depletion random variables. The result this an approach that enables the rapid comparison of spares policies and identification of an optimal policy for a target mass.

The performance of the analytical method was first compared to the simulation results. In aggregate, the results of the analytical approach were comparable to the simulation results. Difference between the two were driven by the need to allocate the

canisters of the backup system in sets to accurately reflect the behavior of the buffer depletion random variable. The impact is especially noticeable at low masses where the first set of canisters must be preallocated to ensure validity of the overall state space whereas the simulation approach can add spares one at a time. Having demonstrated that the analytical approach is able to give comparable results for a fixed set of random variable parameters, the PSO is then used to optimize the spare policy at three allocation masses. For the two lower masses, policies different from the baseline and lazy policies defined in the simulation are found to be optimal. At the highest mass, the repair on failure strategy is found to be optimal which suggests if mass is not particularly constrained or if vary high PoS is valued above all else, the baseline policy should be used. This results also shows that there is not one optimal strategy across all allocation masses, so the planned maintenance strategy may need to be adjusted as the definition of the system is refined. It also validates the need to consider sparing policy as early as possible.

In general, the analytical method will be preferred when quick results are needed, or if many different options are under consideration. The analytical method was conceived as a tool to find optimal sparing policies and it excels at this task relative to the simulation-based approach. It can also be used in lieu of the simulation if firm policy rules have yet to be developed but the modeler has a general sense of what is desired and can express those as simple random variables. This could give relatively quick results and help identify components or spares that drive the analysis as part of a sensitivity analysis.

As discussed in the first chapter, the objective of this research was to develop methods capable of including sparing policy along with system configuration in the estimation of functional availability and associated spares allocations. To meet this

objective, novel contributions were developed across the two methods created to address sparing policy. For the simulation-based approach, the EMAT tool was updated to enable other policies to be tested, including the new Lazy policy. The simulation was also improved through the implementation of the MKP optimization method which allowed optimal spares allocations to be identified in a much more efficient fashion. As for the analytical method, advances were made in the description of the rules for generating the state network diagram using a computer routine and by the implementation of ghost states to eliminate a limitation of SMP. Additionally, the MKP optimization was updated to utilize SMP results to generate allocation which also improved on the previous approach of presenting all possible combinations and tracing the Pareto front. Finally, the analytical approach was used to optimize for sparing policy which has not previously been done. In all, the research objective of this work was met through the two methods develops using several unique contributions.

5.1 Possible Improvements

With a foundation for investigating sparing policy with probability of sufficiency and the associated spares allocations established, further areas for experimentation and investigation are now available to be explored. For the simulation-based approach, new optimization approaches could be explored with new sparing rules sets. For the analytical method, there are opportunities to improve runtimes to allow for exploration of larger state spaces and to try new optimization schemes for the policy parameters to better align the optimizer with the parameter space and underlying shape of the solution space.

The simulation approach does not have large areas for improvement as there is no obvious way to implement the policy searching capability enabled by the analytical method. As discussed in Section 3.1.2, different optimization approaches could be tested if desired or if it aligns better with the larger context of the analysis being performed. Additionally, there is an opportunity to test optimal policies found by the analytical method and to convert them into a rule space. The process of doing this would force the translation of random variable parameters into specific crew behaviors so that the policy could be implemented in the discrete event simulator. Also, the simulation code is a strong candidate for parallelization since many successful simulations are necessary to get the desired results. Refactoring the code base to enable this is another avenue for exploration.

The analytical approach has more opportunities for future improvement and study. The bulk of these opportunities can be binned into two areas, state space management and optimization.

Like any Markov model-based problem, the analytical problem is sensitive to state space explosion. While modern computing capabilities can mitigate this to some degree, there are still areas where improvement can be pursued. Important to this whole discussion, is that the time intensive step of the analytical approach is the calculation of the difference distribution f and its corresponding CDF, F . The difference distribution is calculated through discrete convolution and truncation to $t \geq 0$. This work improved on previous implementations [11] by utilizing circular convolution using FFT/IFFT with element-wise multiplication in the frequency domain as opposed to linear convolution of the built-in MATLAB function which demonstrated slower performance. Even with this change, this calculation dominated the runtime of the analysis. Currently, all difference distributions

are calculated sequentially driven by the fact that a difference distribution at a lower depth in the graph describing the state space may be a function of difference distribution at a higher depth. Some parallelization may be possible by dividing the graph into subgraphs and simultaneously calculating difference distributions across multiple threads. If this improvement could be implemented in a pseudo-recursive manner, spawning new subgraphs for parallelization as higher depth difference distributions are found, it could greatly increase the depths that could be permitted in the state space which could more accurately represent some simulation-based policies like the lazy strategy.

Additionally, some additional performance could be harvested by identifying difference distributions that are not recalculated for each particle or population generation. This is unlikely to be the bulk of the distributions as the parameters describing repair and buffer depletion random variables are being varied to find the optimal policy and the need for the difference distributions is the non-memorylessness nature of the lognormal distribution describing these transitions. As the breadth of the state space grows due to increased elements in the system, the attractiveness of this improvement will grow as there will exist more failure-only difference distributions that do not require recalculation each step and justify the overhead necessary to find these transitions in the automatic state generation step and to track during the difference distribution calculation step. Overall, improving the runtime of the analysis gives the opportunity to explore policies for more complex state spaces.

Improving the runtime of the analysis of a candidate set of parameters can also improve the tradespace of optimizers available to use in finding optimal spares policies. The PSO implemented in this work was adequate to find optimal spare policies but without

the seed of candidate policies, it tended to get trapped in local optima. This is a known issue with PSO and can be improved at the margins by changing the constants w , c_1 , and c_2 in Equation 14 which dictate the relative contributions of the previous velocity and particle and global bests to the new velocity. Values for the constants other than the ones used have been found to be better for certain problems but not enough was known about the shape of the underlying objective function to give confidence that one selection would be better than another. When looking for alternate optimization approaches, the following characteristics of this problem should be taken into consideration:

1. The overall shape of the underlying function is not known but does have a non-negligible number of local optima.
2. The parameters being optimized are continuous.
3. The parameters are inputs into lognormal (or potentially other) random variables, so changes should not be assumed to have a linear impact.
4. Some parameters are the mean and others are the standard deviation so behavior on a parameter-to-parameter level should not be assumed to be similar.
5. The cost of determining the PoS for a given population is non-trivial for large state spaces, so parameter sweeps are similar are not feasible.

One other idea would be to use some subset of results to generate a simplified representation of the optimization function to identify promising subregions to use the PSO. Two possible approaches to this are a response surface or a neural network. For either one, the initial generation of results should capture the entire allocation mass to PoS relationship rather than on a single target mass. Allocation mass should become an input

to the simplified representation. Once a few promising seeds are found, the PSO population can be initiated with them.

Overall, improving the methods set forth in this research should increase the complexity of systems that can be represented a model bringing additional information into the system design process earlier than is currently available. Having better information earlier should benefit designs for long duration crewed missions with restrictive mass constraints, few abort options, and high reliability requirements by providing realistic allocation estimates to achieve the desired outcome. In turn, this information will allow informed solution architecting and provide better insight into the necessary investments to ensure mission accomplishment.

5.2 Areas of Future of Investigation

Beyond improving the underlying methods presented in this work, other strategies from the general area of supportability research could also be included in future work utilizing this research. These approaches include additive manufacturing, cannibalization, and preventative maintenance. Additionally, the potential emergence of super heavy launch vehicles, exemplified by SpaceX's Starship, have the potential to impact the constraints that previously bounded proposed crewed Mars architectures. All of these represent interesting and valid lines of investigation.

5.2.1 Additive Manufacturing

In this context, additive manufacturing refers to the creation of spare parts using a machine that builds the component from bulk material or materials usually using some sort

of layer deposition process. Promising research into the development of low- and micro-gravity additive manufacturing systems for space systems is on-going. The idealized result would be universal commonality where spares mass is reduced to the additive manufacturing machine mass and the input material stock, so each kilogram of spares mass remains uncommitted until the need is presented.

Generally, additive manufacturing introduces some additional considerations that need to be accounted for. First, the time to repair will need to include the time to print the part but sparing policy as defined here is still a relevant concern. One additional near-term consideration is whether the additive manufacturing machine can produce enough of the spares inventory to reduce overall mass needs such that the mass of the additive manufacturing machine, or machines, is offset. Some parametric work [18] has shown that such a trade-off is viable under certain conditions. The methods developed by this research can demonstrate if such an exchange is possible for the specific system configuration under consideration. Additionally, the methods could test hybrid approaches that allocate some spares for expected failures with the additive manufacturing system reserved for subsequent failures.

For the methods shown, the underlying generation of spares demand will not change since they are driven by changes in state driven primarily by failure and repair actions, but additive manufacturing will change how PoS is calculated. For the simulation approach, this could be done in a few different ways. One approach would be to collapse all the spares able to be manufactured into one spare and handle it on a mass basis; for example, how many trips require up to 23kg of additive manufacture mass. Alternately, the separate spares could be kept as is and ORs could be employed in the calculation of POS_{sim} .

In either approach, the MKP subproblem will likely become more complex. The analytical approach will need similar considerations and, in that case, the conversion to mass approach is likely to be more tractable rather than keeping the type of spare separate. In either case, it is possible to envision a path to including additive manufacturing as part of the spares allocation.

5.2.2 *Cannibalization*

Cannibalization is the use of the components of strings that normally provide internal redundancy to be utilized as additional spares in other locations of the system. Usually, this is proposed as an alternative to repairing the string that is subsequently cannibalized. This approach is the inverse of the discussion from Section 3.3.2.2 that discussed additional strings and some of the inefficiencies of adding more and more strings.

From a sparing policy perspective, decisions surrounding cannibalization are very similar to ones already considered. For example, under what conditions should a string be cannibalized? Are they driven by time in the mission? Or, by the types of components that can be spared by the remaining parts of the string? Also, is there a meaningful difference between waiting for a string to fail before cannibalization and active cannibalization of a functioning string? These types of questions are an extension of the question of when to initiate repair and the methods proposed should be suited to help answer them.

To address cannibalization in either the simulation or analytical method, the modeler would need to start by defining the alternative or reduced configuration the system will be in once cannibalization occurs. In the simulation-based approach, a rule defining when to switch to this new configuration needs to be implemented and tested using the

approach described in this work. For the analytical method, a separate state network diagram representing the alternate configuration would need to be described, and then solutions for both would need to be tested over a sweep of time ranges summing to the overall mission duration to find the best time to switch over. Alternately, a combined state network diagram may be possible if carefully constructed, though a cannibalization sink state in the pre-cannibalization state network diagram may accomplish a similar function without complicating some of the renewal and buffer depletion logic and assumptions. For either the simulation or analytical approaches, the modeler will need to accurately account for the reduction in allocation needs provided by the cannibalized string and account for the fact that cannibalization of the same string may yield different spares since a different component may fail to make that string available for cannibalization.

5.2.3 Preventative Maintenance

Preventative maintenance is an action taken to help ensure a functioning system remains functioning. Some of these actions require using mass to complete the task, such as replacing filters. While these actions are often part of keeping a system operating, the inclusion of this type of maintenance is not the best candidate for the methods in this work. Typically, preventative maintenance actions are prescribed at a fixed rate, and if that is all that is known, mass owing to preventative maintenance should be allocated as a separate mass from the spares allocation.

Some preventative maintenance action could be included if some conditions are met. First, a model for how the failure rate of a component changes must be generated. Generally, both methods in this work should be able to handle a non-constant failure rate.

Future modelers would also have to contend with how to classify the preventative maintenance event or action in the simulation or SMP, respectively, and propagate the decision making. Reflecting the decision to initiate maintenance and impact of the maintenance would have characteristic of other behaviours in each model but may be separate enough to require additional development. However, if those conditions are met, then preventative maintenance for certain components could be included in the overall sparing policy of the system.

5.2.4 Super Heavy Lift Launch Vehicles

SpaceX's Starship is an exemplar of a new class of launch vehicle being developed that hope to dramatically reduce the cost of launching mass to orbit by delivering more mass per launch than any previous rocket. The importance of sparing policy in this work is premised on the idea that mass will be at an extreme premium for early crewed missions. It is fair to consider whether the emergence of these types of launch vehicles will eliminate the need to consider sparing policy as part of mission development.

Part of the answer to this question will be dependent on how the additional mass to orbit capability is utilized as part of an overall mission architecture. While a technical marvel, the technology underlying Starship is not such a departure from current rocket technology to drastically change the expected gear ratios for a Mars missions. This means that mass at the top of the mission stack will still need a lot of additional mass to push it around. SpaceX has not released a detailed design of early crewed Mars mission, opting instead to talk about Starship utilization in a steady state colonization scenario [28]. Other work [29] hints some of the additional performance may be used to grow crew size from

six to 10-20. If that choice is made, all accommodations and supporting systems would need to grow. Such a decision could constrain mass in other areas, including sparing mass, which would reintroduce sparing policy as a key consideration.

In other words, the design of a mission architecture for early crewed missions to Mars that utilize Starship should use methods that estimate expected crew survival informed by expected system configuration and spares allocation. The methods in this work are good candidates to support this type of analysis and have the added benefit of explicitly considering sparing policy. If Starship performance is insufficient to remove all mass constraints, then considering sparing policy may still be important. It is reasonable to expect that this constraint will not be fully relaxed, because there will be competing priorities, such as increased crew size, the grow the size of the mission to more than fill the payload fairing of the much larger launch vehicle.

REFERENCES

- [1] Drake, B. G. (ed.). *Human Exploration of Mars Design Reference Architecture*. NASA-SP-566, 2009.
- [2] Rapp, D. *Human Missions to Mars*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
- [3] Young, D., and Wilhite, A. A Resource Allocation Method for Achieving Optimal Reliability in a Lunar Architecture. 2007.
- [4] Brooks, C. G., Grimwood, J. M., and Swenson, L. S. *Chariots for Apollo: The NASA History of Manned Lunar Spacecraft to 1969*. Courier Corporation, 2012.
- [5] Drake, B. G. *Reference Mission Version 3.0 Addendum to the Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team*. NASA/SP-1998-6107/ADD, 1998.
- [6] Bell, T. E., and Esch, K. “The Space Shuttle: A Case of Subjective Engineering.” *IEEE Spectrum*, Vol. 26, No. 6, 1989, pp. 42–46. <https://doi.org/10.1109/6.29339>.
- [7] Hamlin, T., Kahn, J., Thigpen, E., and Lo, Y. Shuttle Risk Progression: Use of the Shuttle Probabilistic Risk Assessment (PRA) to Show Reliability Growth. 2011.
- [8] Kline, R. C., and Bachman, T. C. “Estimating Spare Parts Requirements with Commonality and Redundancy.” *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, 2007, pp. 977–984. <https://doi.org/10.2514/1.28072>.

- [9] Jones, H. Ultra Reliable Space Life Support. 2012.
- [10] Stromgren, C., Terry, M., Cirillo, W., Goodliff, K., and Maxwell, A. Design and Application of the Exploration Maintainability Analysis Tool. 2012.
- [11] Owens, A. C. *Quantitative Probabilistic Modeling of Environmental Control and Life Support System Resilience for Long-Duration Human Spaceflight*. Thesis. Massachusetts Institute of Technology, 2014.
- [12] Pyke, R. "Markov Renewal Processes: Definitions and Preliminary Properties." *The Annals of Mathematical Statistics*, Vol. 32, No. 4, 1961, pp. 1231–1242. <https://doi.org/10.1214/aoms/1177704863>.
- [13] Pyke, R. "Markov Renewal Processes with Finitely Many States." *The Annals of Mathematical Statistics*, Vol. 32, No. 4, 1961, pp. 1243–1259. <https://doi.org/10.1214/aoms/1177704864>.
- [14] Warr, R. L., and Collins, D. H. "A Comprehensive Method for Solving Finite-State Semi-Markov Processes." *arXiv*, 2012.
- [15] Owens, A. C., and de Weck, O. Automated Risk and Supportability Model Generation for Repairable Systems. 2015.
- [16] Messinger, M., and Shooman, M. L. "Techniques for Optimum Spares Allocation: A Tutorial Review." *IEEE Transactions on Reliability*, Vol. R-19, No. 4, 1970, pp. 156–166. <https://doi.org/10.1109/TR.1970.5216436>.

- [17] Siddiqi, A., and de Weck, O. L. “Spare Parts Requirements for Space Missions with Reconfigurability and Commonality.” *Journal of Spacecraft and Rockets*, Vol. 44, No. 1, 2007, pp. 147–155. <https://doi.org/10.2514/1.21847>.
- [18] Owens, A. C. *Multiobjective Optimization of Crewed Spacecraft Supportability Strategies*. Ph. D. in Space Systems. Massachusetts Institute of Technology, 2019.
- [19] Sherbrooke, C. C. *Optimal Inventory Modeling of Systems*. Springer US, Boston, MA, 2004.
- [20] Kellerer, H., Pferschy, U., and Pisinger, D. *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [21] Maxwell, A. J., Wilhite, A., and Ho, K. “Spare Strategy Analysis for Life Support Systems for Human Space Exploration.” *Journal of Spacecraft and Rockets*, Vol. 58, No. 5, 2021, pp. 1394–1405. <https://doi.org/10.2514/1.A34849>.
- [22] de Souza e Silva, E., and Ochoa, P. M. *State Space Exploration in Markov Models*. 1992.
- [23] Abate, J., and Whitt, W. “Numerical Inversion of Laplace Transforms of Probability Distributions.” *ORSA Journal on Computing*, Vol. 7, No. 1, 1995, pp. 36–43. <https://doi.org/10.1287/ijoc.7.1.36>.
- [24] Nunn, W. R., and Desiderio, A. M. *Semi-Markov Processes: An Introduction*. 1977.
- [25] Kennedy, J., and Eberhart, R. *Particle Swarm Optimization*.

- [26] Hassan, R., Cohanin, B., de Weck, O., and Venter, G. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. 2005.
- [27] Helwig, S., Branke, J., and Mostaghim, S. “Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization.” *IEEE Transactions on Evolutionary Computation*, Vol. 17, No. 2, 2013, pp. 259–271. <https://doi.org/10.1109/TEVC.2012.2189404>.
- [28] Musk, E. “Making Humans a Multi-Planetary Species.” *New Space*, Vol. 5, No. 2, 2017, pp. 46–61. <https://doi.org/10.1089/space.2017.29009.emu>.
- [29] Heldmann, J. L., Marinova, M. M., Lim, D. S. S., Wilson, D., Carrato, P., Kennedy, K., Esbeck, A., Colaprete, T. A., Elphic, R. C., Captain, J., Zacny, K., Stolov, L., Mellerowicz, B., Palmowski, J., Bramson, A. M., Putzig, N., Morgan, G., Sizemore, H., and Coyan, J. “Mission Architecture Using the SpaceX Starship Vehicle to Enable a Sustained Human Presence on Mars.” *New Space*, Vol. 10, No. 3, 2022, pp. 259–273. <https://doi.org/10.1089/space.2020.0058>.