NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COURSE OF ACTION ANALYSIS FOR THE U.S. ARMY'S 2035 MODERNIZATION

by

David R. Black

June 2021

Thesis Advisor: Hong Zhou
Second Reader: Jenifer R. McClary

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE COURSE OF ACTION ANALYSIS FOR THE U.S. ARMY'S 2035 MODERNIZATION	**5. FUNDING NUMBERS**
6. AUTHOR(S) David R. Black	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) The Research and Analysis Center, Monterey, CA 93943	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

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

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The United States Army Futures Command (AFC) is comparing multiple modernization levels of various programs in order to prepare for military operations in 2035 and beyond. This thesis supports that effort by analyzing a data set of three modernization level possibilities for 68 programs. There are 2.78 nonillion possible combinations in this data set, and the current methodology requires an analyst to select a single combination, and then calculate the overall benefit score. The time to complete this calculation is over 1 second, so by using the current methodology it would require 8.8*10^22 centuries to compute. Therefore, this thesis started with two objectives: (1) develop a faster method for calculating the benefit score for a singular combination of programs, and (2) develop a methodology for comparing multiple scores to each other at once. While in pursuit of the first two objectives, the programming Julia proved to be exponentially faster for singular calculations, resulting in the ability to view nearly 9,000 scores within one second, all while using CPU encoding. These faster calculations led to the development of a third objective: compare the speed and accuracy of a machine learning (ML) algorithm. The third objective resulted in speeds 40 times faster than the CPU model, but with a relative error of 1.3%.

14. SUBJECT TERMS Julia, machine learning, Army Futures Command, TRAC	**15. NUMBER OF PAGES** 65
	16. PRICE CODE

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

NSN 7540-01-280-5500

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

COURSE OF ACTION ANALYSIS FOR THE U.S. ARMY'S 2035 MODERNIZATION

David R. Black
Major, United States Army
BS, U.S. Military Academy, 2009
MS, Missouri University of Science and Technology, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by: Hong Zhou
Advisor

Jenifer R. McClary
Second Reader

Wei Kang
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The United States Army Futures Command (AFC) is comparing multiple modernization levels of various programs in order to prepare for military operations in 2035 and beyond. This thesis supports that effort by analyzing a data set of three modernization level possibilities for 68 programs. There are 2.78 nonillion possible combinations in this data set, and the current methodology requires an analyst to select a single combination, and then calculate the overall benefit score. The time to complete this calculation is over 1 second, so by using the current methodology it would require 8.8*10^22 centuries to compute. Therefore, this thesis started with two objectives: (1) develop a faster method for calculating the benefit score for a singular combination of programs, and (2) develop a methodology for comparing multiple scores to each other at once. While in pursuit of the first two objectives, the programming Julia proved to be exponentially faster for singular calculations, resulting in the ability to view nearly 9,000 scores within one second, all while using CPU encoding. These faster calculations led to the development of a third objective: compare the speed and accuracy of a machine learning (ML) algorithm. The third objective resulted in speeds 40 times faster than the CPU model, but with a relative error of 1.3%.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AFC Army Futures Command

CPU Central Processing Unit

DM Dependency Matrix

GPU Graphical Processing Unit

ML Machine Learning

MPI Message Passing Interface

RCM Resource Coefficient Matrix

SOBJA Scaled Objective A

SOBJB Scaled Objective B

TRAC The Research Analysis Center

TOB Total Operational Benefit

TSVV Theater Specific Value Vector

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank my wife (my unofficial third reader) and daughter for their love, support, and understanding as I worked on this from home during our 15-month long "shelter in place."

Additionally, I would like to thank Dr. Hong Zhou, who served as my thesis advisor, for her inspiration and motivation, Dr. Jeremy Kozdon for teaching me the Julia programming language, Major Jenifer McClary for keeping me from straying away from the objectives, and Captain Kert St. John for being a sounding board.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

The goal of this thesis is to support the Army Futures Command (AFC) course of action analysis as they prepare for operations in the year 2035 and beyond. For background, the "Army Futures Command (AFC) leads a continuous transformation of Army modernization in order to provide future warfighters with the concepts, capabilities and organizational structures they need to dominate a future battlefield" [1]. AFC currently analyzes mixtures of modernization levels for various programs and yields a Total Operational Benefit (TOB) score as a result. However, the problem with AFC's current model is twofold: (1) scale and (2) speed.

1.1 Scale

The provided data set from AFC looks at two theaters with the same 34 programs in each theater. However, the 34 programs in each theater are independent of other theaters and other programs. The fact that there are two theaters with 34 programs each is equivalent to saying there are a total of 68 programs for modernization analysis. The separation of theaters will be important to note during the background chapter. Additionally, the scale is reduced in this thesis by limiting each program to three discrete modernization level possibilities: (1) 0% modernization (keep the legacy equipment), (2) 50% modernization (update half of program's fleet), and (3) 100% modernization (update the entire program). Therefore, with 3 levels to choose from, and 68 total programs, the total number of possibilities is 3^{68} (2.78 nonillion). For perspective, 3^{68} divided by the approximate number of stars in the Milky Way (400 billion), is still 6.95×10^{22}.

1.2 Speed

AFC's current algorithm utilizes the R programming language. This approach entails an end user manually selecting the modernization level of all 68 programs and then running the program to get the TOB score. When timing the algorithm alone, and excluding the time it takes the end user to determine which levels to select, the process takes over one second.

Therefore, when excluding the end user's time to select each option, and just using the algorithm to compute every one of the 3^{68} possibilities, it would take 8.8×10^{22} centuries to compute. Furthermore, AFC's current approach is not holistic. Since the end user must select what they think might be a good combination of modernization levels across the 68 programs, and then calculate a new combination to compare, the end user is forced to compare two singular values with little reference of how they compare to any other option. These challenges provide the genesis for this thesis.

1.3 Research Questions

This thesis started with two objectives: (1) investigate/develop a faster method for calculating the TOB score for a singular combination of programs, and (2) develop a methodology for comparing multiple scores to each other at once. However, during the research, a third objective emerged. The reasons for this development are discussed in greater length in chapter two, but the third objective is to develop and investigate the speed and accuracy of a machine learning (ML) algorithm using the Julia programming language. The assumption is that a ML algorithm will reduce future calculation times by eliminating the various intermediate calculations. These intermediate calculations are discussed in the next chapter.

CHAPTER 2:
Background

This chapter explains the necessary background information the reader needs in order to set the stage for the remaining chapters. A discussion of programming languages, the data set used, and the underlying equations used by AFC will be discussed.

2.1 Programming Languages

There are several reasons this thesis uses the Julia programming language. First, AFC's current algorithm uses the R programming language, which is an interpreted language, much like another common programming language named Python. This means that R and Python both use an interpreter to read and then send code to the Computer Processing Unit (CPU) for execution. Conversely, Julia is a compiled language, meaning Julia code is executed directly on the CPU, giving the programmer more control over hardware aspects like memory management and CPU usage [2]. This observation is the original reason Julia was chosen: compiled programming languages are faster than interpreted programming languages.

Secondly, Julia is an open source and free language, which sets it apart from MATLAB (another common programming language) or other programs requiring a subscription. This allows AFC to continue to research this thesis without the constraints of a MATLAB subscription. It also means there are various packages the end user can download to expand the capability set of the code over time, much like R and Python. The current AFC code requires multiple packages to be downloaded and runs the risk of version upgrades. Therefore, a goal of this thesis is to minimize the use of external packages and keep the coding as pure as possible. This approach requires the most basic level of coding to avoid future syntax errors and is achieved by requiring only one package, the "Linear Algebra" package.

Thirdly, Julia is very user friendly for parallel computing. For example, future work on this topic could include using Julia's GPU and Message Passing Interface (MPI) capabilities in order to calculate multiple scores at once. To elaborate, Julia allows the programmer to compute hundreds of TOB scores on different CPUs and/or GPUs at the same time, and

then sort and compile those scores for the end user. However, for this thesis, the assumption is made that AFC does not want to use additional hardware, such as servers and multiple computers. Therefore, with this assumption, coupled with Julia's adequate computation of thousands of TOBs in seconds, GPU and MPI capabilities are not leveraged in this thesis.

Finally, Julia provides a first-class array implementation that does not expect programs to be written in a vectorized style for performance. In Julia's online manual [3], it states that "Julia's compiler uses type inference and generates optimized code for scalar array indexing, allowing programs to be written in a style that is convenient and readable, without sacrificing performance, and using less memory at times." As will be seen in future sections, these computations all utilize matrix operations. Therefore, Julia's abilities to optimize matrix operations are highly valuable.

For these reasons, Julia is the programming language of choice for this thesis. Additionally, all portfolio and program computations are conducted in Float64, meaning they are done using double precision. A faster but less accurate approach of Float32 (single precision) is used to accelerate the ML algorithm.

2.2 Data Set

As mentioned in Chapter 1, the data set for this thesis comes from AFC. More precisely, The Research and Analysis Center of Monterey, which is a subordinate unit to AFC, utilizes this data for research purposes and provided this set. The specifics of the data set are obfuscated to protect the sensitive nature of the data. Therefore, it is sufficient to explain that the data set is comprised of two theaters, which are actual geographical regions of the world, but to prevent any misuse of the information, the theaters are called Theater A and Theater B.

2.2.1 Portfolio vs Program

Within each theater there are nine portfolios. The portfolios are identical between the two theaters, and an example of a portfolio title is "Future Vertical Lift." This broad title encompasses three or four programs, such as a new helicopter program, and an unmanned aerial vehicle program. The programs are identical across theaters, just like the portfolios, and there are a total of 34 programs in each theater. The labeling convention is the same as

the portfolios, so the programs are labeled A through AH. Table 2.1 shows which programs belong to each portfolio.

Portfolio	Programs
A	A, B, C, D
B	E, F, G, H
C	I, J, K, L
D	M, N, O, P
E	Q, R, S
F	T, U, V, W
G	X, Y, Z, AA
H	AB, AC, AD, AE
I	AF, AG, AH

Table 2.1. Programs within Portfolios.

2.2.2 Objectives

AFC is focused on 11 objectives, as shown in Figure 2.1. The objectives are similar to the six warfighting functions (movement and maneuver, intelligence, fires, sustainment, and protection), as outlined in Army Publishing Directorate 3-0 [4]. However, AFC's specific objectives are not titled or described in this data set or thesis for security purposes. It is important to know that these 11 objectives, which are identical across the two theaters, are weighted differently in terms of their value to achieving overall success within each theater. This will play a crucial role in differentiating between the best portfolios and programs for each theater when the results are discussed.

2.3 Underlying Equations

The following subsections provide an explanation of both AFC's equations and some modifications to those equations. Also, there are many acronyms that are used to describe concepts and they serve as variables or inputs for the Julia coding. The format of the description below is bottom up, meaning the first derivation consists of the user inputs and the final outcome (Total Operational Benefit) is described last.

2.3.1 Theater Specific Value Vector

The first calculation is the Theater Specific Value. In AFC's algorithm this is an individual score for each of the 11 different objectives. However, by leveraging Julia's matrix arithmetic capabilities these 11 scores are treated as a vector, and I will, therefore, refer to them as the Theater Specific Value Vector (TSVV). The TSVV is best viewed as a possible combination of modernization levels for a set of programs within a theater, and how much value that set of modernization levels contributes to each of the 11 objectives. The following subsections provide a more thorough explanation of the three components of the TSVV, to include elaboration of the 11 objectives. In short, the three components are the Resource Coefficients, Dependencies, and Capability Values. Again, as a common theme in this thesis, Julia's matrix arithmetic capabilities are leveraged, resulting in the development of the Resource Coefficient Matrix (RCM), Dependency Matrix (DM), and Capability Value Vector (CVV). The equation for the TSVV is:

$$[TSVV]_{(obj,thtr)} = [RCM]_{(obj,pro)} \times [DM]_{(pro,pro)} \times [CVV]_{(pro,thtr)} \tag{2.1}$$

For the given data set, the RCM is a (11 objectives x 34 programs) matrix for each theater, the DM is a square matrix of (34 programs x 34 programs), and the CVV is a column vector of (34 programs x 1 theater). Thus, the TSVV is (11 values x 1 theater).

Resource Coefficient Matrix (RCM)

As stated in Section 2.2.2, AFC has 11 objectives, labeled 1 through 11, and these objectives are identical across the two theaters. The range of these values are between 0 and 1, with 1 representing the highest operational benefit contribution to a particular objective. Therefore, the overall matrix is captured in an 11 x 68 matrix, or 11 x 34 for each separate theater. Table 2.2 provides an excerpt for programs A, B, and AH in Theater A.

Objective	Program A	Program B	Program AH
1	0.0250	0.2875	0.3500
2	0.4000	0.4375	0.3500
3	0.0000	0.1500	0.3500
4	0.0000	0.2625	0.3500
5	0.0250	0.6313	0.8500
6	1.0000	0.8750	0.6500
7	0.0000	0.0000	0.0000
8	0.1250	0.1250	0.2000
9	0.0000	0.1375	0.1000
10	0.0000	0.2000	0.0500
11	0.0000	0.0875	0.2000

Table 2.2. Resource Coefficient Table (Theater A, Programs A, B, and AH).

When looking at Table 2.2, the score for Objective 6 and Program A is 1. This is the largest score compared to any other score within Program A. Therefore, it is interpreted that Program A supports Objective 6 more than any other objective. By referencing the entire data set, which is not provided due to security concerns, it is seen that Program A and Z have the same score of 1. Therefore, it is true that Program A supports Objective 6 more than any other objective, but it cannot be said that Program A supports Objective 6 more than any other program supports the same objective.

Additionally, for Program B, we see that it also contributes to Objective 6 with a score of 0.8750. Since Programs A and B are both in Portfolio A, it is highly probable that Portfolio A is more closely associated with supporting Objective 6 than any other objective.

Lastly, Programs A, B, and AH all have an operational resource coefficient of 0 for Objective 7. So it is assumed that one of the 34 programs supports this objective, otherwise the objective is not a focal point for AFC.

Dependency Matrix (DM)

In Chapter 1 it is stated that all the programs are independent of one another. This is a simplification of the complexities TRAC is working through. In reality, there is a probability

that a dependency exists between programs, and it is feasible that multiple portfolios, or programs across portfolios, have a dependency on one another. An example might be helpful: assume a network communication system is a program, it stands to reason that a helicopter, or entire portfolio like "Future Vertical Lift," will be impacted by the network communication program. However, the network program most likely won't depend on the helicopter program, or the vertical lift portfolio. Therefore, a dependency must exist to ensure the modernization level of one program (e.g. network communication) accurately depicts the impact on other program modernization levels, even though they are in different portfolios.

This complexity of dependencies that TRAC is working through, means dependencies are beyond the scope of this thesis. Therefore, a place holder called the Dependency Matrix (DM) is inserted into the code for future work. An example of future work would be to conduct a sensitivity analysis of this set of values, but for the time being, the place holder is an identity matrix of size 34. If, during TRAC's continuing research, it is determined that there is a dependency between two programs across the two theaters, it would drastically complicate this work. The resulting TVVS would require a DM of size 68, the theaters could not be calculated separately, as will be mentioned in the following chapters, and the computational effort would dramatically increase, given that the TVVS would become a row vector of size 68.

Capability Value Vector (CVV)

In order to rate the value each program provides to the US Army, and not just a particular objective, TRAC leverages subject matter expertise from over 80 studies to define a capability value for each program between 0 and 100 [5]. For example, assume Program A is the helicopter example, and the 0% modernization of Program A, in Theater A, is 0. The 50% modernization of the same program in the same theater is 51.25, and the capability value of 100% modernization is 65 (see Table 2.3). The interpretation of these values is somewhat revealing. To explain: the decision to not upgrade the helicopter (0% modernization) means the legacy equipment provides no value for future warfare in Theater A. Yet, 50% modernization, comparable to maintaining 50% of the legacy equipment and equipping 50% of formations with new equipment, increases the value to 51.25. Furthermore, a complete fielding of new equipment (100% modernization) only returns a value of 65. Clearly, the

values of 0, 50, and 100% and the capability values are not linearly dependent. Otherwise, the expectation would be that an increase in modernization yields a proportional increase to the capability value. For example, doubling the modernization level from 50% to 100% should result in a capability value increase that doubles as well. Since this does not exist, it is assumed the functions which define the relationship between the modernization level and capability value is continuous rather than discrete. However, as mentioned before, the three modernization levels result in 3^{68} possibilities; if we increased it to 100 modernization levels, it would increase the number of possibilities to 100^{68}.

Fortunately, the modernization of these programs is required to be somewhat discrete. Using the helicopter example, again, the program is referring to the modernization of a fleet of helicopters. Therefore, to modernize the helicopter program of 25 helicopters to 50%, half of the legacy equipment would be replaced, which would either require 12 or 13. As such, discrete levels do exist and would become much more complicated problem, one beyond the scope of this thesis, if more than 0%, 50%, and 100% levels are used. Future work could look at more than three modernization levels, or at the possibility of continuous functions, which represent those levels. For now, Table 2.3 shows show the capability values for Theater A, Portfolio A (Programs A through D).

Program	0% MOD	50% MOD	100% MOD
A	0.000	51.250	65.000
B	23.750	61.250	77.625
C	24.333	48.333	58.111
D	24.333	27.444	30.555

Table 2.3. Capability values for Theater A, Portfolio A.

Of note, from Table 2.3, AFC states many legacy programs still provide a capability value. For example, referring to Table 2.3 it is seen that 0% Modernization results in 0 value for Program A, 23.750 for Program B, 24.33 for Program C and D. This suggests that 0% modernization, also known as relying exclusively on legacy equipment, still provides a certain value in a future conflict. Again, this observation could be used to investigate more trends or conduct future research beyond this thesis. For example, is the difference in Program D's scores of 24.333, 27.444, and 30.555 worth the cost differences of these

three modernization levels, especially compared to Program A's higher difference of 65.000 and 0.000 when looking at 100% or 0% modernization? The key takeaway for this thesis, though, is that a change in these values will impact the values in the Scaled Objective, which is discussed next. Therefore, it is important to recognize that these values only change periodically when new studies, war-games, or simulations provide updated results. This allows for the hard coding of this information in the Julia code rather than pulling from a file or treating them as variables. However, when they do change, it could drastically alter the final outcome.

2.3.2 Scaled Objective (SOBJ)

The TSVV calculated in the previous section can result in values beyond 100%. As a result, the values are scaled with respect to both the maximum and minimum scores that could be achieved if all programs were modernized to 100% or 0% respectively. The resulting value is called the Scaled Objective (SOBJ). Since each theater has different objective weights, the Scaled Objective values will be different. Therefore, each theater has its own function. SOBJA denotes the Scaled Objective for Theater A, and SOBJB denotes the Scaled Objective for Theater B. Equations 2.2 and 2.3 are the calculations of the SOBJA and SOBJB. Each component of the two Scaled Objectives is elaborated in the following subsections.

$$SOBJA = \frac{(TSVV - MiPVA)}{(MaPVA - MiPVA)} \times (OMaA - OMiA) + OMiA \tag{2.2}$$

$$SOBJB = \frac{(TSVV - MiPVB)}{(MaPVB - MiPVB)} \times (OMaB - OMiB) + OMiB \tag{2.3}$$

Minimal Possible Value (MiPV)

The Minimal Possible Value for Theater A (MiPVA) is a column vector of 11 values, which represents the minimum possible TSVV for a given objective.

Maximum Possible Value (MaPV)
Opposite of the MiPV, the Maximum Possible Values for Theater A and Theater B represent the highest TSVV for a given objective.

Objective Maximum (OMa)
This column vector of size 34 represents the maximum score of an operational measure when all programs are modernized to 100%.

Objective Minimum (OMi)
Similarly, the Objective Minimum is a column vector of size 34, representing the minimum score of an operational measure using the legacy equipment, or 0% Modernization.

2.3.3 Operational Benefit for Theater A and B (OBA and OBB)

The Operational Benefit for Theater A and B (OBA and OBB) is the value of the most interest for each theater. It represents the quantity of benefit given the levels of modernization selected. As for how the OBA is calculated, it is comprised of two components: (1) SOBJA/SOBJB and (2) the Weighted Operational Score. The SOBJA/SOBJB has already been discussed in the previous section, and the Weighted Operational Score is discussed in more detail in the next subsection. For reference, this is the equation for the OBA/OBB:

$$OBA = [SOBJA] \times [WOPSA] \tag{2.4}$$

$$OBB = [SOBJB] \times [WOPSB] \tag{2.5}$$

At this point, the current algorithm returns singular values, such as 8488.87, which is difficult to interpret. Therefore, this value is scaled in coding. For example, the highest possible OBA under the R coded algorithm is 8488.87, which is, obviously, the highest level of modernization for every program. However, the end user could have difficulty separating how big the loss in benefit is between two different sets of modernization levels. What is the magnitude of impact of 8487.87 vs 8488.87?

To make the values more understandable, the highest possible value, and all subsequent

values, are divided by the maximum. Therefore, for the OBA of 100% modernization across all programs, the score is 100. Any reduced modernization level of any of the 34 programs would, therefore, result in a score out of 100. Not surprisingly, a 50% modernization across all programs returns an OBA or OBB of nearly 50.

Weighted Operational Score for Theater A and B (WOPSA and WOPSB)
The Weighted Operational Score for Theater A and B (WOPSA and WOPSB) denotes an assigned value, as determined by 20+ surveyed operational and institutional general officers (GOs) [5]. This is a column vector of 11 values, and is developed by input the GOs provide, in terms of the impact each objective has on overall operations within a theater. For example, a score of 7 for Objective 1 versus a score of 9 for Objective 5 within theater A is interpreted as the GOs consensus that Objective 5 is more important to overall success within Theater A than Objective 1. As such, these values are only periodically changed as priorities over time change.

2.3.4 Total Operational Benefit (TOB)

This is the final value of the computation and pulls in all the programs, all the theaters, and the end user's selection of modernization levels. It is a singular value, in which the larger it is, the better it is for the US Army. It is comprised of four components: (1) the Weighted Theater Score for Theater A (WTSA), (2) the Weighted Theater Score for Theater B (WTSB), (3) the OBA, and (4) the OBB. The WTSA and WTSB are explained in depth in the following subsection, but the equation for the TOB is as follows:

$$TOB = (WTSA \times [OBA]) + (WTSB \times [OBB]) \tag{2.6}$$

Weighted Theater Score for Theater A and B (WTSA and WTSB)
These values denote the end user's perspective on the importance of each theater as a whole. The values are between 0 and 1, and sum to 1. For example, a WOPSA of 0.5 requires a WOPSB of 0.5, and this represents that the two theaters are equally as important. As a different example, a WOPSA of 0.3 requires a WOPSB of 0.7, and is interpreted as Theater B being more than twice as important than Theater A for the U.S. Army.

CHAPTER 3:
Portfolio Analysis

As mentioned in the background chapter, the OBA and OBB are calculated separately prior to incorporating the decision maker's WOPSA/WOPSB values. While an initial analysis of the OBA and OBB results in only $2 * (3^{34})$ (33 quadrillion) values, rather than the total 3^{68} values, this initial analysis proves to be very useful, as shown in the following sections.

3.1 Mathematical Model

The crucial component of the mathematical model is the creation of a function that calculates the OBA , a separate function that calculates the OBB. However, there are a few things to highlight about the creation of the functions. One observation is that Julia utilizes an inverse matrix multiplication when the division symbol is used. This is handy for calculating the SOBJA and SOBJB since it decreases the number of calculations. Also, by hard coding values into the Julia coding, no information is pulled from a database, or CSV file, like Microsoft Excel. Instead, the values are compiled when the code is executed. This further leverages Julia's abilities by creating 1-dimensional arrays, which accelerates the computation time by preventing the CPU from having to communicate across multiple computer programs. A drawback of this approach is that sensitivity analysis would need to be coded directly into the Julia file, rather than allowing an analyst to manipulate a simple CSV file. However, the expectation is that these values will only periodically change, since they are a combination of a General Officer Panel and Subject Matter Experts, who determined these values over multiple iterations and studies.

Now that a function has been created for the OBA, and separately the OBB, we have simplified the process by requiring only 34 inputs, which are the 34 choices of modernization levels, for each function. These levels are represented as values "1, 2, and 3" and correspond to the three levels of modernization, as listed in the 1-dimensional arrays. For example, no modernization of a program is represented as a "1" while 100% modernization of a program is represented as a "3." Therefore, to calculate the OBA when all portfolios are modernized, except Portfolio A, the inputs are as follows:

$$THEATERA(1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3) \tag{3.1}$$

The inputs must be in alphabetical order, so when the function in Equation 3.1 is used to calculate "all portfolios are modernized, except Portfolio B," the inputs are "3,3,3,3,1,1,1,1,3,3,3,3...". As a result of this approach, we can calculate all 33 quadrillion OBA scores by creating a "34-level nested loop." It is important to keep Table 2.1 in mind, because it provides the referencing for which programs are in each portfolio. A simple mistake of some portfolios having four programs versus three programs is a possibility.

The function to this point, is being run on an Intel Core i5-4590 CPU @3.30GHz processor. It clocks in at approximately 9,000 OBA scores calculated per second. Therefore, with 16.5 Quadrillion scores possible for the OBA, it would take 581 centuries to calculate every possibility. Therefore, to massively reduce the required number of computations, we don't need to calculate more than 21 scores. The following subsections discuss why these 21 scores are selected and provides an analysis of the OBA scores. However, up until this point, the 9,000 scores are calculated, which are a component of the overarching equation, in one second. Comparing this to the overall equation used by AFC, which takes one second for a single computation, we are starting to see the time savings. In reality, AFC is not attempting to calculate every combination, but this methodology allows them to compare 9,000 combinations to one another at once. Therefore, it is clear that this new code is much faster and now provides a near simultaneous comparison of different OBA options.

3.2 Theater A

Digesting 16.5 quadrillion possibilities for the OBA is unrealistic. Therefore, an initial analysis of modernizing all but one portfolio to 100% was conducted. The following graphic provides the 21 scores, which show the impact of this potential decision. The green line represents modernizing all portfolios to 100%; therefore, all analysis is based as a percentage of the best possible solution. Also, the yellow line represents all portfolios being modernized to 50%, and the red line represents keeping the legacy equipment (aka 0% modernization).

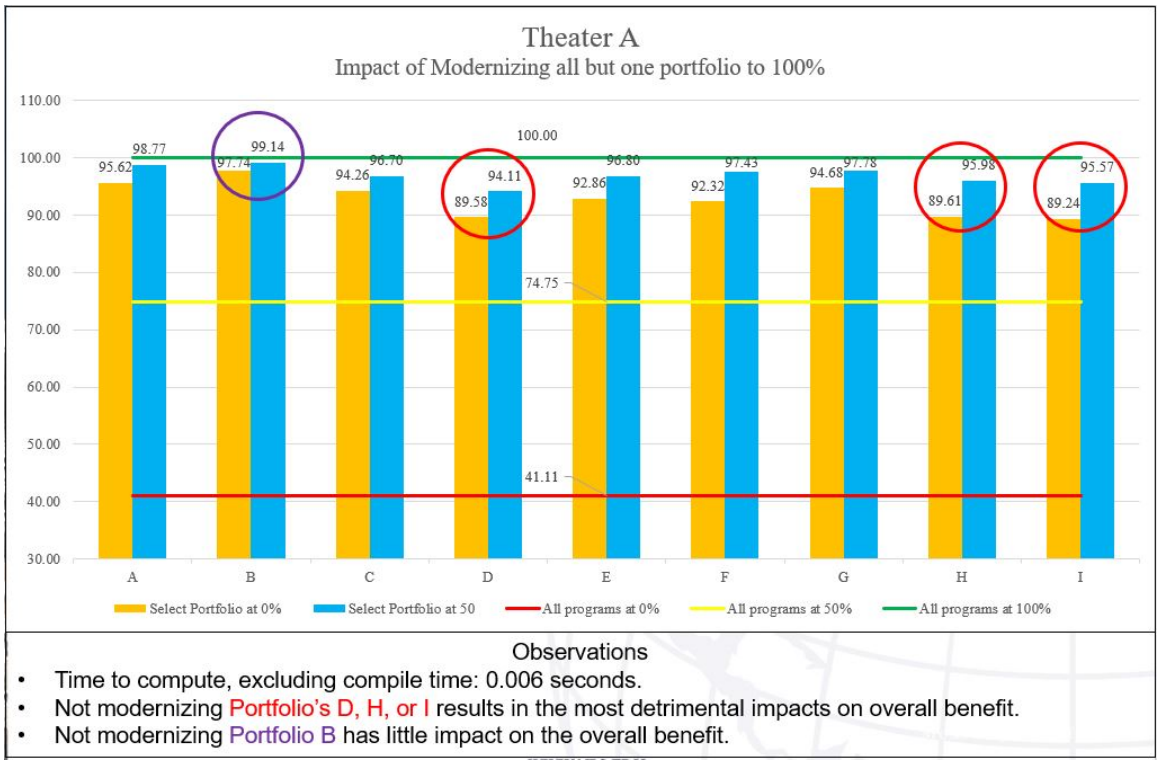

Figure 3.1. OBA Portfolio Analysis

From this chart, we see several note-worthy items. First, not modernizing Portfolio's D, H, or I (highlighted with red circles) results in the most detrimental impacts on the OBA. This approach immediately shows the three most important portfolios within a theater, and is determined by these three portfolios having the biggest gaps from the green line. For illustration, note that when Portfolio I is not modernized at all, we see the biggest decrease to the OBA with a score of 89.24. If Portfolio I is modernized to 50%, while all others are at 100%, we see a score of 95.57, which is higher than if Portfolio D is the only one constrained to 50% modernization. This means the three most critical Portfolios for Theater A are D, H, and I. As such, the recommendation to AFC is that, in a mutually exclusive environment between Theaters A and B, the most critical portfolios in Theater A are D, H, and I. This identification should, therefore, trigger further investigation into these portfolios to better understand whether all programs are critical contributors, or if it's exclusive to specific programs.

Secondly, not modernizing Portfolio B has little impact on the overall benefit. In fact, the blue bar shows a score of 99.14 when Portfolio B is modernized to 50%, or 97.74 when not

modernized at all. Therefore, if there was one portfolio where money could be saved, this would be a good place to begin investigating.

Finally, a visualization of the most important Portfolios listed in descending order is warranted. Table 3.1 provides this by depicting the average OBA when a Portfolio is the only one not being fully modernized.

Portfolio	Average OBA Percentage
D	91.84
I	92.41
H	92.79
E	94.83
F	94.87
C	95.48
G	96.23
A	97.20
B	98.44

Table 3.1. Average OBA for Not Modernizing a Portfolio to 100%

As an example, note that Portfolio A results in two possible OBAs in the bar chart, 95.62 and 98.77. When averaging these scores, a value of 97.2 is found, which means, on average, it is worse to not modernize Portfolio A in Theater A than it is to not modernize Portfolio B in Theater A.

3.3 Theater B

In Figure 3.2, the green line once again represents all portfolios being modernized to 100%. However, you will see that the yellow and red lines have different values in this theater compared to Theater A. This is the first finding that is important. It tells us that a 50% modernization in Theater B yields a 76.5% return, compared to a 74.75% return in Theater A. Meaning an overall flat line modernization of 50% in Theater A is more detrimental than doing the same process in Theater B. Or stated differently, Theater B can assume more risk in modernizing to only 50% in all portfolios. However, the same cannot be said for

keeping the legacy equipment. A decision to not modernize at all will be more detrimental in Theater B, compared to Theater A.

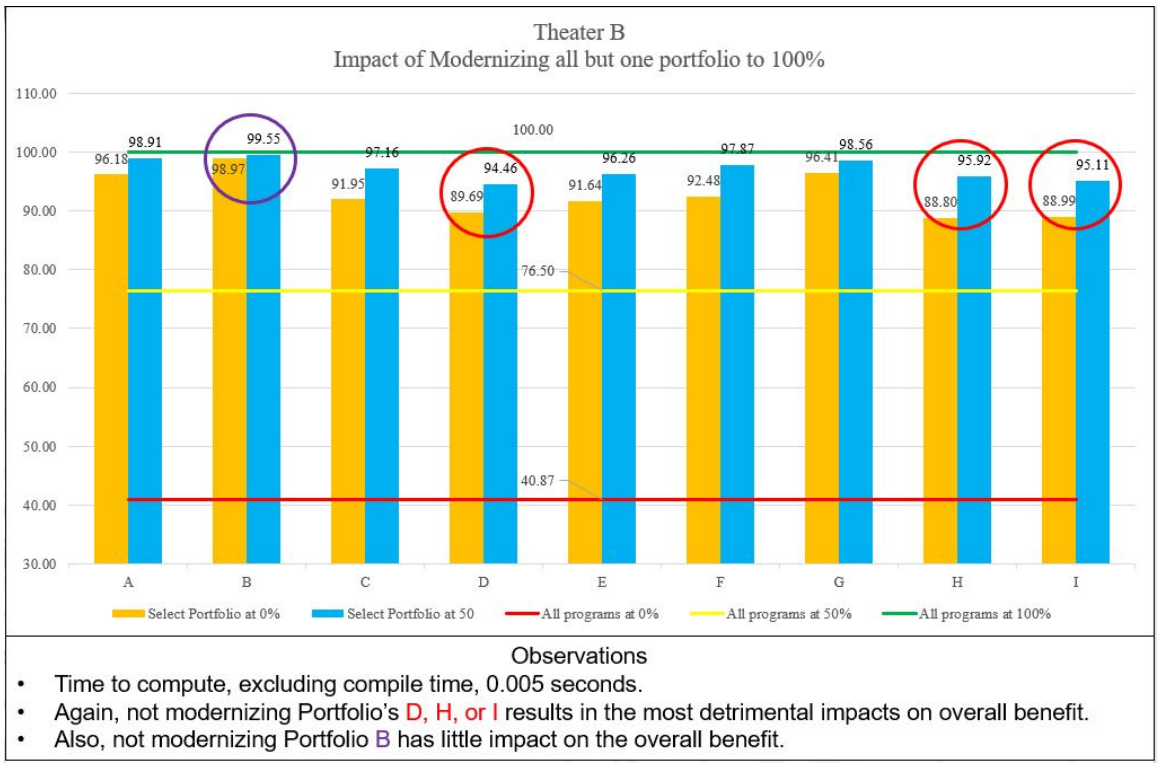

Figure 3.2. OBB Portfolio Analysis

Performing the same analysis as Theater A, we see that the most critical portfolios are D, H, and I. They are highlighted by the red circles. Notice that these are the same portfolios we identified in Theater A as having the most detrimental impact if not modernized. Also, the least critical portfolio is B, as it was in Theater A. However, the order and magnitude of each portfolio to each theater is not the same. Table 3.2 shows this difference by listing the order and magnitude for Theater A next to the order and magnitude for Theater B. The values are averaged as they were in Table 3.1.

Theater A	Percentage		Theater B	Percentage
D	91.84		I	92.05
I	92.41		D	92.08
H	92.79		H	92.36
E	94.83		E	93.95
F	94.87		C	94.55
C	95.48		F	95.18
G	96.23		G	97.49
A	97.20		A	97.55
B	98.44		B	99.26

Table 3.2. Comparison of Portfolio Analyses for Theaters A and B

From Table 3.2 we see that not only is Portfolio D the most important to Theater A over Theater B, but it is also more detrimental to Theater A than any portfolio loss in Theater B. However, because the values are averages between 0% and 50% modernization, we see that Portfolio I is actually the most important portfolio for Theater B. Similarly, Portfolio B is the least important portfolio in both theaters. In fact, it provides very little benefit (less than 2%) in either theater, and should, therefore, be evaluated in terms of overall procurement.

3.4 Summary

The computations in this chapter only took a few seconds. Yet, those computations quickly highlighted the percentage loss between not modernizing a portfolio relative to modernizing all other portfolios to 100%. This led to the following key observations: (1) not modernizing at all results in a more detrimental impact to Theater A than it does to Theater B, (2) Portfolios D, H, and I are the most important for both Theater A and Theater B, and (3) Portfolio B is the most expendable of the 9 portfolios, and warrants additional investigation. However, this is a macro level analysis of many combinations, and we can now shift our focus by diving deeper into how each program affects a portfolio, the OBA/OBB, and the TOB. This will be the focus of the next chapter.

CHAPTER 4:
Program Analysis

While an analysis of the impact of portfolio modernization is useful and only took fractions of a second to compute, a more powerful analysis is easily conducted by comparing the impacts of not modernizing single programs. As stated in the Portfolio Analysis Chapter, these values are part of the 59 scores that took 0.02 seconds to compute. By modifying the existing code to export these values into excel, we can dive deeper into program analysis for each theater.

4.1 Mathematical Model

The mathematical model for Program and Portfolio Analysis is the same, other than iterating over programs instead of portfolios. For example, in Portfolio Analysis we used the Equation 4.1 for Portfolio A:

$$THEATERA(1,1,1,1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3) \tag{4.1}$$

Now that we are looking at a single program, we have only a single "1" with 33 inputs as "3." For example, assume we are looking at modernizing Program A a 0%, and all others are 100%, then the function call is shown in Equation 4.2.:

$$THEATERA(1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3) \tag{4.2}$$

Therefore, as mentioned in the Portfolio Analysis Chapter, while we use the same functions for the OBA and OBB, we must now calculate at least 35 scores. The reason for comparing 35 scores is that altering 34 programs to 0% individually creates 34 bars on a bar-chart, with the 35th score being the 100% modernization of all programs. This reduction of looking at just 0% allows for a more easily digested comparison than the 69 scores that would be required to show 0% and 50% modernization.

4.2 Theater A

The OBA is depicted for 35 combinations in Figure 4.1. The first combination is 100% modernization of all programs in Theater A, and is represented by the green line. Subsequently, every score in the bar chart represents an OBA that is a percentage of the total possible score. For example, if we modernize every program to 100%, except Program B, the OBA is 98.09% of the maximum possible value. Also, to aide in connecting portfolios and programs, the red lines depict which programs belong to each portfolio.

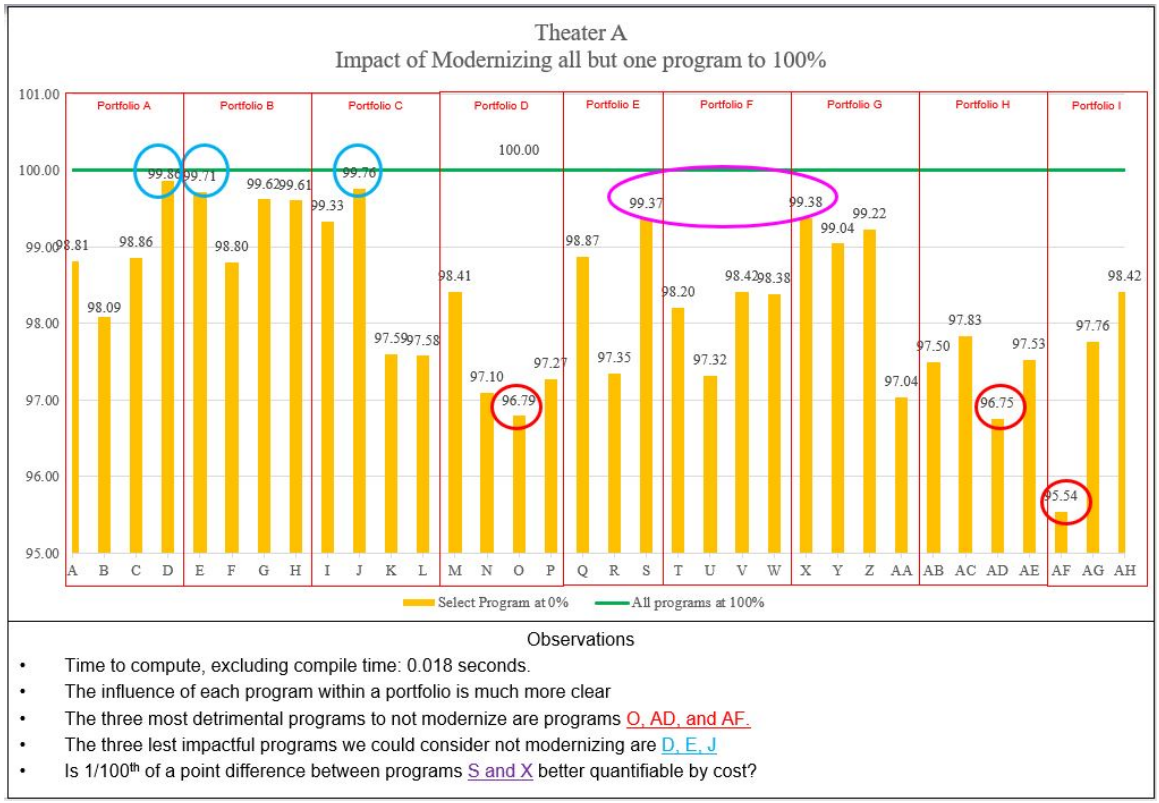

Figure 4.1. OBA Program Analysis

From Figure 4.1 we see several note-worthy areas for investigation. First, assuming the focus is on the three most detrimental programs, like we did with the portfolios, we identify programs O, AD, and AF and highlight them with red circles. It is not surprising that these three programs correspond to the most critical portfolios mentioned in the Portfolio Analysis chapter, namely Portfolios D, H, and I. What is revealing, though, is that we stated Portfolio D is the most critical. Yet, the most critical program is not in Portfolio D, but rather it is Program AF in Portfolio I. Therefore, to add strength to our previous statements,

observations, and analysis, we must state the most critical program for modernization in Theater A is Program AF, but the most critical Portfolio is D.

Next, the three least important programs for Theater A are D, E, and J. These are highlighted with the cyan circles. An example of this observation is that not modernizing Program D still results in 99.86% of the total possible score. Therefore, future analysis is warranted to determine if this is an area of potential cost savings, possibly even an opportunity to shift financial resources to programs that are more critical, like Program AF.

Thirdly, Program S has a score of 99.37, while Program X has a score of 99.38. How much does a 1/100th of a point matter? Arguably, our approach of identifying the top three, and bottom three programs, with red and cyan circles, is useful to determine the direction of further research. However, taking it to the next step and highlighting small differences, such as that between Program S and X, should also trigger additional investigation. This is where this thesis provides a foundation for future work, specifically, this thesis provides an analysis of ranking benefit, but how do these scores correlate in terms of magnitude for financial constraints?

Finally, as we did for the Portfolio analysis, a ranking of the most critical programs in Theater A are provided below. However, to be concise, just the top ten are listed. For example, as mentioned, Program AF is the most critical program because it causes the biggest reduction in the OBA when not modernized. Therefore, it is listed at the top of the table.

Portfolio	Average OBA Percentage
AF	95.54
AD	96.75
O	96.79
AA	97.04
N	97.10
P	97.27
U	97.32
R	97.35
AB	97.50
AE	97.53

Table 4.1. Average OBA for Not Modernizing a Program to 100%

4.3 Theater B

Similar to the graphic for Theater A, the graphic below provides a visualization for Program Analysis in Theater B. The green line represents full modernization within Theater B, and all subsequent values represent a percentage of the green line when a program in question is not modernized. The red lines depict the program and portfolio association.

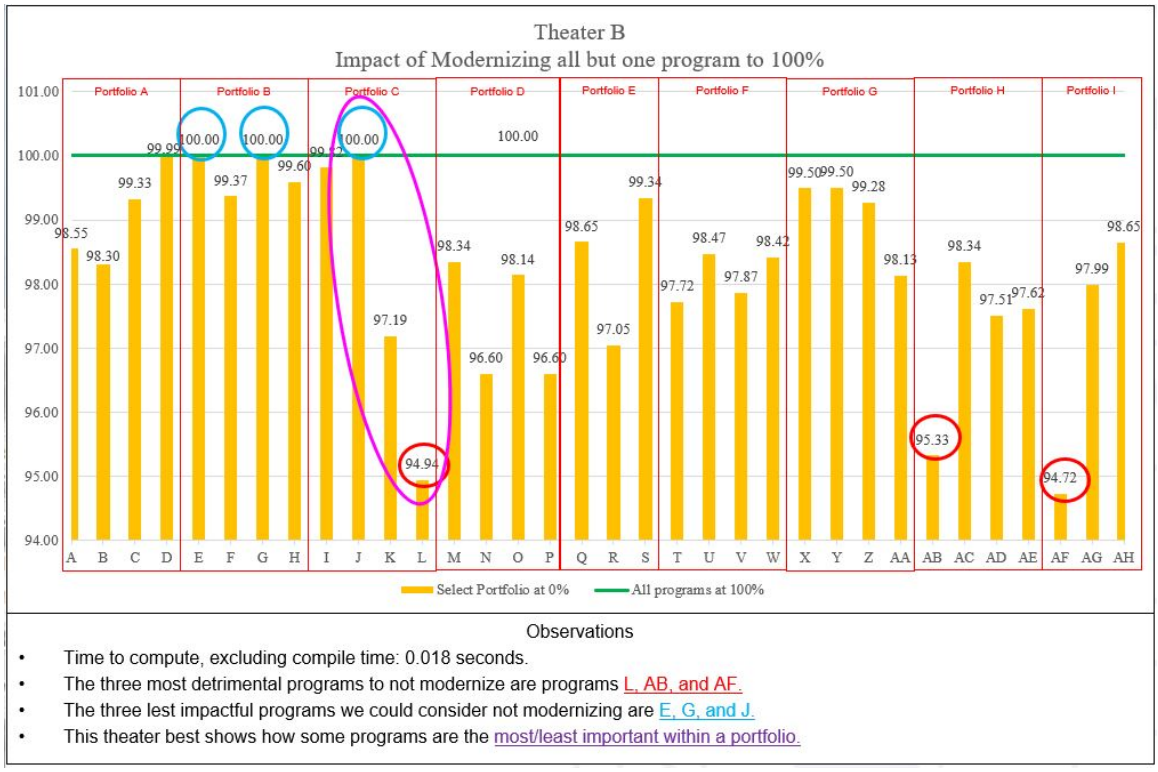

Figure 4.2. OBB Program Analysis

Different from Theater A, the most critical programs are AF, L, and AB, respectively. These are highlighted by the red circles. Also, the least critical are programs E, G, and J, which are different from Theater A. In this graphic, the purple line depicts a radical difference between two programs within the same portfolio. Program L is one of the top three most important for Theater B, and the most important in Portfolio C, but Program J provides no benefit across the theater of the portfolio. Yet, both these programs are in the same portfolio.

Furthermore, a surprising finding is that the three least critical programs create such a small drop in the percentage. In fact, the drop is so small that the value of 100 for Programs E, G, and J is actually just an effect of rounding to the nearest one hundredths place. This warrants additional investigation, since these programs provide relatively no benefit and therefore, needs to be explained to justify procurement.

4.4 Comparison

We can now compare Theater A and B to see which programs are the most important for the TOB. This is best visualized by combining the bar charts into a single graphic, which is provided in Figure 4.3:

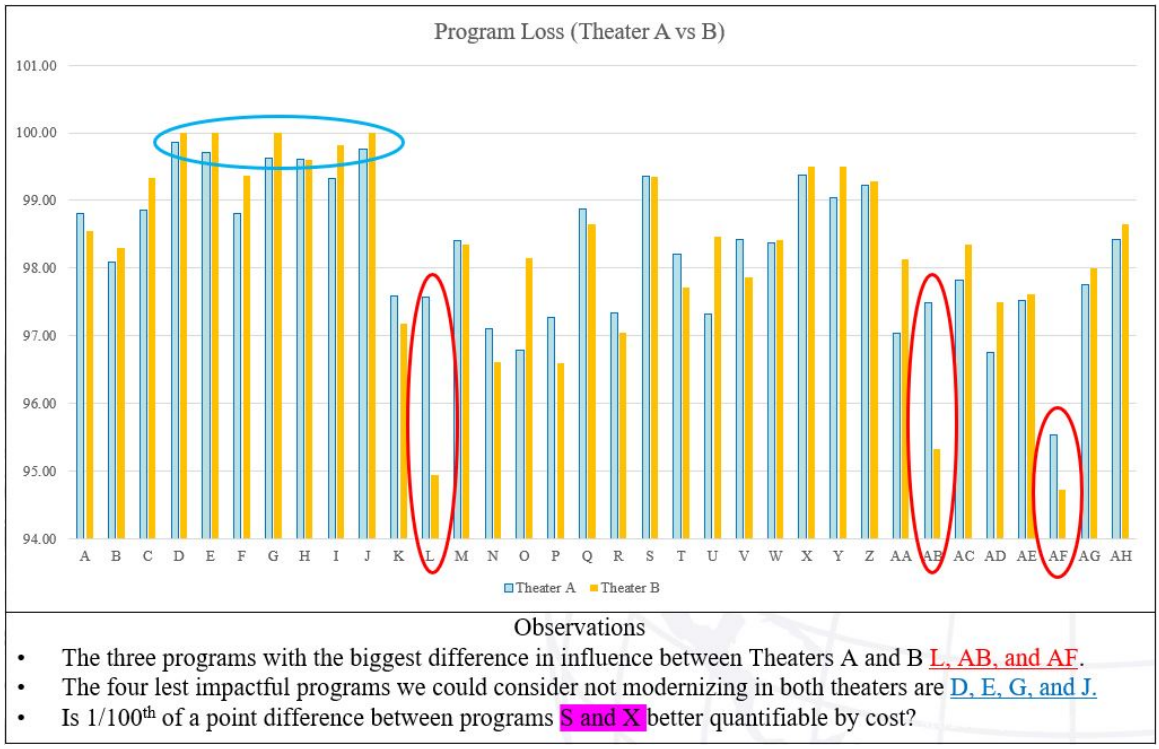

Figure 4.3. Program Loss as a Theater Comparison

By combining these charts into one, we see that Programs D, E, G, and J have a commonality of providing very little benefit for both theaters. This is represented by the blue oval. It is interesting to note that these four programs do provide some benefit to Theater A, but are still the least critical programs across both theaters.

Additionally, from Figure 4.3, we can see that Program AF creates a major loss, if not modernized in both Theaters A and B. However, there is a major difference in the degree of change between the two theaters and that gap is, similarly, reflected in Programs L and AB. These three observations are highlighted with the red ovals. For example, the loss of Program L for Theater B is extremely significant (reduction to below 95%). Yet it is only marginally significant (reduction to around 97.5%) if removed from Theater A.

4.5 Programs of Interest

From these observations of Programs D, E, J, L, AB, and AF, we will define them as Programs of Interest. The purpose of this label is to advocate for a deeper dive in determining the practicality of modernizing a program that provides little to no benefit. Conversely, a highly influential program may benefit from a surge in resources so that the positive return is gained more quickly. That analysis is beyond the scope of this thesis. However, with our data set we can more clearly articulate the interaction between a set of programs. For example, the graphic below shows the effects of the three most critical programs (L, AB, and AF).

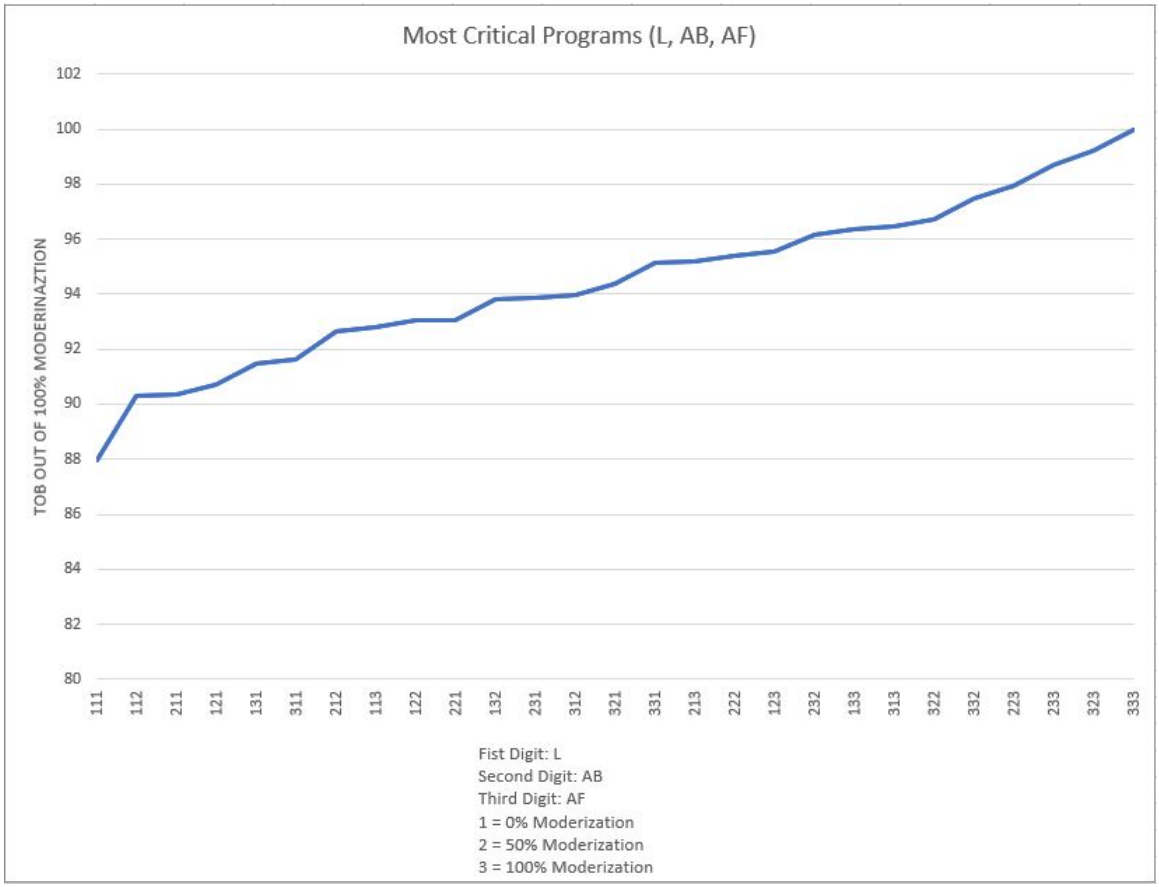

Figure 4.4. Most Critical Programs

To understand Figure 4.4, recall that a value of "1" in the coding for OBA and OBB means 0% modernization, "2; means 50%, and "3" means 100%. In addition, as stated in the key, the first digit on the x-axis represents Program L, the second digit is Program AB, and

the third is program AF. This means that "111" represents no modernization of any of the three programs in either theater, but a full modernization of all other programs across both theaters. The first y-value is approximately 88%, the lowest return as a percentage of possible modernization. The subsequent values along the x-axis are sorted, such that the next higher percentage is listed, which shows that Programs L and AB would be 50% modernized, and Program AF would be 50% modernized. Therefore, the purpose of Figure 4.4 is the variation of TOBs as we iterate through every permutation of Programs L, AB, and AF. A key take away is that a single modernization of any of the three programs creates a significant jump in the overall percentage. Therefore, if AFC can only choose 1 of the three programs, they should choose Program AB, since it returns a higher score than any other combination of 2 x 0% and 1 x 50% modernizations.

This analysis serves as an example of the level of deep dive analysis that can occur with simple manipulations of "for loops" and well-defined functions. Another more practical example is the analysis of an individual portfolio. For example, if AFC wants to better understand how the four programs of Portfolio A create variances in the overall score, we can quickly iterate through the 81 possibilities, as shown in Figure 4.5. In this figure, the values are sorted so that the highest score is listed as "Combination Number 1" and the lowest possible score is "Combination Number 10." The values under the columns of A, B, C, and D denote the level of modernization required to achieve the score and are color coded for easy of visibility. The second highest score comes from a 100% modernization of Programs A, B, and C, and are colored green, with a 50% modernization of Program D which is colored yellow. This allows the reader to see trends, such as Program B is green in the top nine scores, with the most variation in Program D. Therefore, Program D is more expendable than Program B in terms of calculating an overall benefit. While this was previously deduced, this deeper dive into all 81 combinations may provide answers for more specific relationships between programs. All 81 combinations are provided in the appendix.

Combination Number	Score	A	B	C	D
1	100.00	100	100	100	100
2	99.96	100	100	100	50
3	99.92	100	100	100	0
4	99.71	100	100	50	100
5	99.69	50	100	100	100
6	99.68	100	100	50	50
7	99.65	50	100	100	50
8	99.64	100	100	50	0
9	99.61	50	100	100	0
10	99.47	100	50	100	100

Figure 4.5. Analysis of Portfolio A

4.6 TOB

As stated in the beginning of this thesis, there are 3^{68} combinations to choose from. However, those combinations did not include the various levels of WTSA and WTSB that are possible. For example, a single combination of programs chosen for Theater A and Theater B have an infinite number of TOBs, since the TOB is of infinite size. For example, the combined score of WTSA + WTSB = 1. However, there are an infinite number of possibilities within that basic constraint. If we state the WTSA and WTSB must be a single decimal place, we significantly reduce the number of possible TOBs to ten, but that is ten TOBs per combination of the 3^{68} possibilities. Therefore, consideration of all possible TOB, WTSA, and WTSB combinations is not practical. However, now that we have narrowed our focus to specific programs, we can compare ten WTSA possibilities for a specific combination of programs to see if anything revealing is found. Figure 4.6 is used to show that comparison.

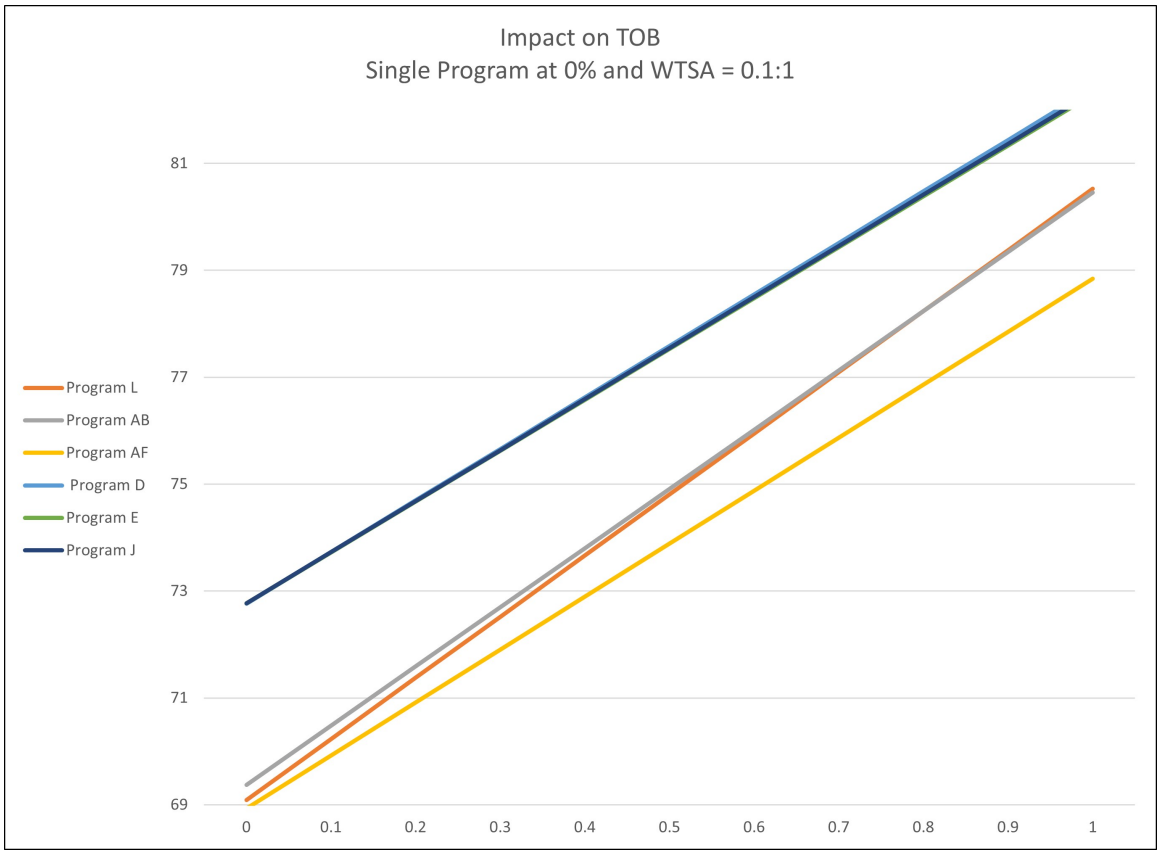

Figure 4.6. TOB Comparison

In Figure 4.6 six lines are depicted, which correspond to the three most critical programs (Programs L, AB, and AF), the three least critical programs (D, E, and J), and compares the TOB of ten different WTSA levels. Keep in mind that WTSA = 1-WTSB, and both WTSA and WTSB are in the range of 0.1 to 1, with discrete computations every 0.1.

Our first observation from Figure 4.6 is that "Program D", "Program E" and "Program E" all look like the same line, with a slight differences in their slopes, as can be seen in the thicker appearance of the line as WTSA approaches 1. Which means their impact on the TOB, as the WTSA/WTSB changes, are basically the same. However, we deduce that the higher the WTSA, the higher the TOB. Meaning Theater A's valuation is highly impactful on the TOB, regardless of the which of the three programs we are looking to delete. Furthermore, there is almost no difference in which program we should delete.

However, the three lower lines in Figure 4.6, corresponding to programs L, AB, and AF,

show a differential in their slopes. This means that as the WTSA increases, the importance, or impact, of program AF is more heavily felt. This is seen by the gradual slope of the line compared to a steep slope. Therefore, as the WTSA increases, Program AF continues to be the most important program, because its loss still creates the biggest loss on the TOB. Finally, the chart shows that when the WTSA=0, the TOB is heavily impacted, regardless of which of the six programs are not modernized, meaning Theater A is clearly the most important theater.

While these findings do not seem monumental on the surface, an evaluation of the coding shows that the TOB for all six programs, at all ten levels, which requires the calculation of the OBA and OBB ten times each, only took 0.2 seconds. Furthermore, this 0.2 seconds included the compile time, and subsequent calculations will prove to be even faster, if such analysis is desired. As a further example of the power of rapid calculations, we can dig into a particular portfolio and see how the possible combinations of individual programs affects the score.

4.7 Summary

In short, thousands of scores can be calculated in rapid succession using basic "for loops," allowing the end user to quickly identify trends, as we have done in this and the previous chapters. By starting at the portfolio level, then moving to the program level, and then considering a range of WTSA possibilities, the analysts and/or end user is able to see these trends in under one second, which is how long the R program took to calculate a single score. Therefore, while much more additional research could be conducted on this data set, with this coding we have answered the questions most likely to be asked: "what programs can we cut, which programs can't we cut, and which theater has the largest impact on the overall benefit." Therefore, our ability to calculate 9,000 scores per second is impressive, but the methodology of determining which scores to analyze first is more, and now achieved.

The next chapter takes a different approach with this data set. The approach is to explore a machine learning algorithm built for Julia called FLUX, and to see how accurate and how fast it is in comparison to our current base model. The purpose of taking this new approach is to eliminate all the intermediate calculations, as outlined in the background chapter. In short, a ML algorithm will take a set of inputs and known outputs, and then

create intermediate equations through back-propagation that are more simplistic for future output calculations.

CHAPTER 5:
Machine Learning Algorithm Analysis

This chapter takes a different approach to solving AFC's problems of speed. Rather than using AFC's current equations to constantly calculate the TOB, a machine learning model takes a training set of known inputs and outputs, and then estimates the outputs of future inputs. This basic summary is covered in more depth in the following sections, which will cover the mathematical model with assumptions, results, and recommendations of this approach. While the original question was to calculate the speed of the ML model, we must also ask "how accurate are the results?" This last question is the final portion of the chapter.

5.1 Mathematical Model

While the intent of using a ML algorithm for this thesis is to increase speed and to evaluate its accuracy, it is noteworthy that the main purpose of a ML algorithm, in general, is typically used for solving systems of nonlinear functions [6]. Which is, potentially, the exact situation this data set provides. For example, as mentioned in Section 2.3, the governing functions for the CVVs are nonlinear. Additionally, the 80+ studies, multiple GO panels, and continuing research are all examples of how this problem could be modeled as a system of nonlinear equations. As such, an approach of using linear regression with various activation functions might prove to be useful, especially as the problem becomes more complex with financial constraints and a dependency matrix. However, using linear regression for nonlinear data will, more than likely, not be 100% accurate. Therefore, future research could look at using various ML algorithms beyond this thesis.

5.1.1 Assumptions

The first observation is that this machine learning task is defined as simple regression, which is the process of predicting a real value for each set of inputs [6]. This leads into the various assumptions we must make, which will simplify the model. The first simplification is that we will only look at the inputs of the modernization level for the 34 programs in Theater A. This was chosen to preclude the WTSA complexities and to provide a base model for comparing simplistic "for loops" to a ML model. In short, the overall intent was to train and test on

the same number of combinations as the base model, rather than requiring 68 inputs and including the variables of the WTSA and WTSB. The outputs, therefore, are the OBA for each combination. With this clarification, the overarching model is best described as a neural network in which various layers are established for the inputs to "pass through" to produce a singular output. These layers utilize weights and bias that are randomly selected, initially. However, it is the user's decision to establish the number of layers, and the number of nodes within each layer. Therefore, based on the previous explanations of the nine portfolios and 11 objectives, the assumption is made and implemented by creating three layers. The first layer consists of the decision of the modernization level of each of the 34 programs, therefore it is represented by 34 nodes with three unique outcomes for each node. This layer is not affected by the back propagation and therefore does not have weights or bias. The second layer consists of nine nodes (representing the nine portfolios), and the third layer consists of 11 nodes (representing the 11 objectives). This assumption is pivotal to the model, since more nodes, and/or more layers, will cause more computations (thus increasing the time for calculations), but provide more weights and bias for alteration during the back propagation of the linear regression.

In terms of the back propagation, the output is not binary, nor is it a set of classification probabilities. As such, the Sigmoid and Softail functions proved to be inappropriate for either layer two or three, and the ADAM optimizer is used instead. ADAM is a "first-order gradient-based optimization" that is "straightforward to implement, is computationally efficient, has little memory requirements, and is well suited for problems in terms of data and/or parameters [7]." Of note though, dozens of options exist within Julia for different optimizers, and a much more thorough analysis could be done to determine the best fit model and/or value for an optimizer. However, through trial and error, a value of 0.1 is deemed adequate. Additionally, the loss function used is the mean square error between the actual output value and the model's output value. Lastly, the number of back propagations is the final portion of the model the user must determine. For this thesis we use 20 back propagations (referred to as epochs in Julia code). This decision is based purely on the speed of calculations for the size of the training set and may not, necessarily, be the best level. However, the results between 1 and 5 epochs was minimal, while an increase to 10 and 20 proved to provide far better results. Very little improvement was seen beyond 20 epochs and took much longer for the overall code to run.

In summary, the ML algorithm entails many assumptions: (1) loss function selection, (2) optimizer function and value, (3) the number of layers, (4) the number of nodes within the layers, and (5) the number of back-propagations. These decisions do not even include the choices for the training and testing sets, which are also determined by the end user and discussed next.

5.1.2 Training and Testing Sets

In order to compare the ML model with the original coding in Chapters 3 and 4, the training and testing sets were compiled to be the same size. Therefore, using an 8-level "for loop," a total of 6,561 OBA scores served as the training set and another 6,561 OBA scores served as the testing set. Both the training and testing sets were developed by random selection of program combinations, and the selection of those combinations is saved so that a comparison can be done on these true values for the testing set compared to the ML predicted values.

5.2 Results

With the stated assumptions and objectives, we can now analyze the speed and accuracy of this ML attempt. In terms of time, the training set took 0.85 seconds to compute, but the model predicted the other 6,561 scores in 0.02 seconds. While this seems amazingly fast, it is important to denote all the time steps. The first time step is the time to compile the equations within Julia for the CPU to make rapid future calculations. This took 0.53 seconds. The second time step was building the training set, which took 0.83 seconds, as just mentioned. The third time step is building the testing set, which, not surprisingly, took 0.82 seconds since it is the same size as the training set. The fourth time step is the back propagation, in which 20 back propagations were conducted. This took 31.39 seconds. The fifth step is the model's prediction of scores, which is the amazingly fast 0.02 seconds. Therefore, the total time was 33.6 seconds. Which means the ML model is able to calculate a training set at the normal speed as in the previous chapters, but it can then predict future scores 41.5 times faster. That is amazingly fast, and this is all done on the same CPU as before. To increase the speed, GPUs or arrays of CPUs/GPUs, like an MPI, could be used to conduct these operations in parallel. However, the ability to calculate nearly a half million scores in one second is more than adequate for this thesis.

In terms of accuracy, the difference between the model's solutions and the testing solutions are depicted in Figure 5.1, which is the difference between the model's OBA and the true OBA for 100 random scores out of the possible 6,561 scores that were calculated. For example, in Combination 1, the model's OBA was 1.5 points higher than the true OBA (model OBA = 98.5 points, true OBA = 97 points).

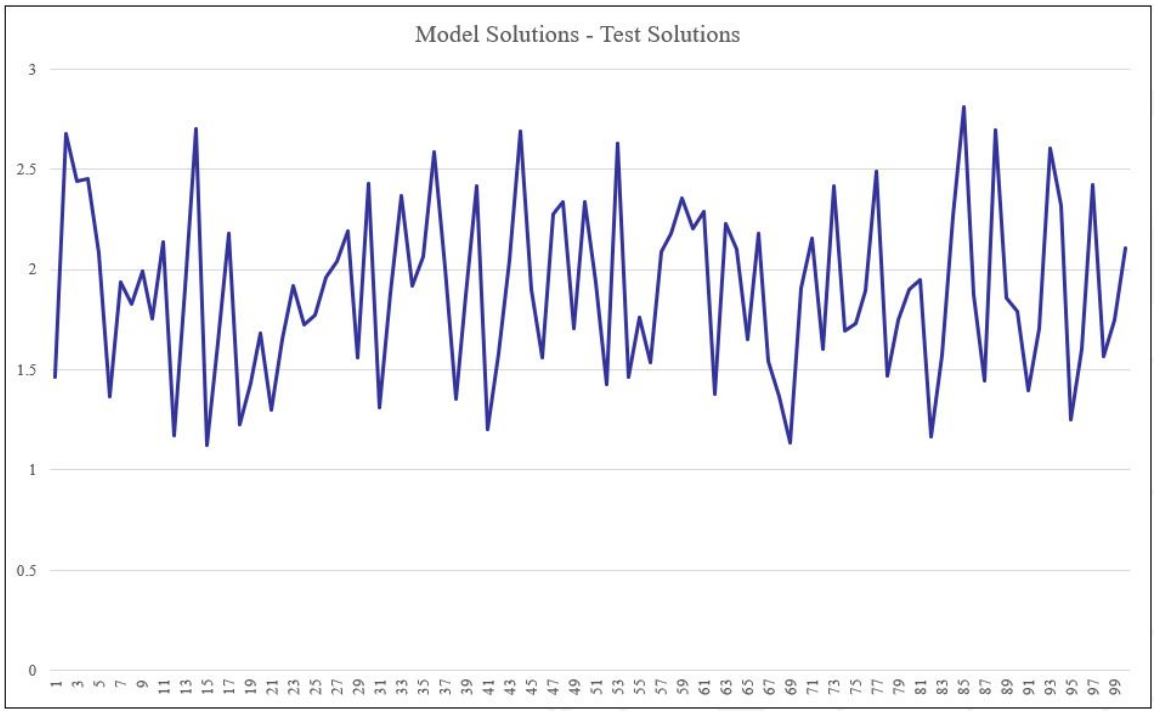

Figure 5.1. Difference of 100 out of 6,561 OBA Scores Using the Machine Learning Model vs. Testing Set Results

The key takeaway from Figure 5.1 is that there is a positive difference between the model solutions and the testing solutions, meaning the current ML model is overestimating for these 100 scores. This could be the result of the programs used for the training set. For example, if the training set uses programs A through H, which can be seen if Figure 3.1, the results will be small impacts to the OBA when they are not modernized. Conversely, if the testing set uses Portfolios D and E, which have a larger impact on the OBA when they are not modernized, then the governing equations for the layers won't be able to adequately compensate for the larger variations. Based on our attempt here, it appears reasonable that our ML model is overestimating the OBA for this very reason. Therefore, a future ML model should use a wide range of programs for the training set, and perhaps more than

6,561 scores should be calculated as the baseline. Additionally, to this point only 100 of the 6,561 scores have been discussed. If we were to test all scores, the graph would look more like a straight line due to the number of values along the x-axis. A better, less visual interpretation is therefore used to quantify the error. That better method is using the relative error, which is calculated using equation 5.1.

$$RelativeError = \left| \frac{ModelAnswers - TestingSetAnswers}{TestingSetAnswers} \right| \times 100\% \tag{5.1}$$

This calculation tells us the percent error of the model in terms of what the output value should be, rather than a total percentage. For example, rather than stating the model is 99% accurate, we state that of the 25 trials conducted, the best relative error is 1.3%, meaning our values are 1.3% of what they should be. Which begs the question "what relative error is acceptable?" Arguably, the speed of getting a 100% correct answer, which is 9,000 TOBs (not just OBAs) is sufficient. Therefore, while this ML model is 41.5 times faster, it produced 1.3% relative error on the best iteration, and shouldn't be relied on for best results.

5.3 Recommendations

This ML model was fast by all accounts (e.g. comparison to AFC's current model and comparison to our CPU computation). However, the accuracy and assumptions made to get to that speed are not ideal and are not useful considering the speed of 100% accuracy using the methodology established in Chapters 3 and 4. Therefore, the current ML model is not recommended for this problem set, but it does highlight some areas of future scientific computation and data analysis for AFC, and perhaps this data set, for purposes other than creating a faster, holistic methodology for addressing the two original problems.

Key areas that could be investigated more are the choice of the training and testing sets, both their sizes in general and relative to each other, and the portfolios and programs used. For example, a random selection of programs from every portfolio could provide a better training set. This is facilitated by the current speed of the Julia algorithm, which is summarized in Table 5.1.

of Programs	Combinations	Time
4	81	0.009 seconds
8	6,561	0.854 seconds
12	531,441	67.545 seconds
16	43,046,721	91.201 minutes
20	3,486,784,401	Approx. 123 hours
68	2.78 nonillion	Approx. 225,622 centuries

Table 5.1. Current and Estimated Calculation Times for TOB

From Table 5.1, we see that the time required is approximately 91 minutes for 16 programs. However, developing a training set that is 20 programs would not only require over 5 days of computation, but the storage of those solutions would be immense, especially in double precision. Therefore, future investigations into training and testing sets must consider the amount of time required to train and validate the model, whereas the current code is sufficiently fast enough without using ML. Consequently, careful consideration should be given for selecting the best of 20 or fewer programs for the training set. Additionally, the selection of quantities for the layers and number of nodes in each layer should be assessed along with the method of "learning" such as our use of ADAM. With all of these recommendations, it leads to a bigger discussion of overall recommendations and future research areas, which will be covered in Chapter 6.

CHAPTER 6:
Conclusions and Future Research

This thesis looked at solving the problems of speed and scale in AFC's current algorithm for calculating a total benefit when potentially modernizing 34 programs across two theaters, totaling 3^{68} possible combinations. The methodology entailed changing the programming language from R to Julia, creating "for loops" to calculate multiple outputs in rapid succession, establishing a metric for comparisons, recommending approaches for analyzing macro-level decisions, such as theater and portfolio level modernization, and analyzing micro-level decisions, such as program modernization afterwards. Additionally, this thesis evaluated one machine learning model in terms of its accuracy and speed. The intent of this chapter is to consolidate and summarize the conclusions and recommendations that have been stated throughout the chapters. Additionally, potential areas of future research are identified.

6.1 Conclusions

AFC's initial request was for an analysis of a "light model" that could quickly evaluate TOBs. Since this may have resulted in a redesign of the underlying equations, this thesis took a different approach by keeping the current algorithm, but adjusting the implementation from the R to Julia programming language. Using Julia proved to be much faster, and it retains the benefits of being free and downloadable on government computers just like R. Additionally, Julia can leverage parallel computing and GPU utilization. Finally, Julia can be run on tablets, which allows an analyst to operate in a more austere environment and still conduct scientific computing. As a result, though R is a widely used program within the analytic community, it might not be the best program for this data set. The first conclusion of this thesis is that Julia is better for rapid calculations of large data sets, and best suited for this type of analysis.

In terms of theaters, Theater A has a greater impact on the TOB than Theater B. This was shown in Section 4.6 by demonstrating the effect of a WTSA ranging from 0 to 1 with WTSB = 1-WTSA. In every case, the TOB is lower when the WTSA is lower. Therefore,

AFC should focus future investigations around modernizing Theater A before Theater B. Additionally, the dependency of the programs and portfolios in Theater A should be analyzed first to ensure the most critical programs in Theater A are modernized as soon as possible. Finally, if 100% modernization is not attainable across all programs, then cuts to Theater B should first be considered pending all other factors to be considered. For example, are there non-quantitative factors such as weather, or integration with host nations, that are not included in the current model?

As summarized in Table 3.2, portfolios D, H, and I are consistently the most critical portfolios across both theaters. Therefore, AFC should focus future research on programs and portfolio dependencies within these three portfolios. The portfolio that creates the least impact is Portfolio B. In fact, the impact is so small that this could trigger an investigation on the entire portfolio. For example, can the entire portfolio be eliminated, or does it have dependencies between other portfolios and programs? Or, is there a better assortment of programs that can better meet the intent of Portfolio B, since it currently provides such a small benefit for predicted operations in 2035? In reality, this assessment is only based on the given data set, which is obscured and does not include non-quantitative values such as a commander's instinct, and is therefore just a starting point.

Program L, AB, and AF create the biggest loss to the TOB when not fully modernized. Therefore, a dependency on these programs, by any other program, should be investigated as soon as possible. Additionally, these three programs create the biggest impact to Theater B, but not to Theater A. This shows that the composition of best assets for the two theaters is not identical, but these three programs share a correlation of importance between the two theaters. Furthermore, programs D, E, G, and J provide almost no benefit to Theater B, and cause the least impact to Theater A. This should be investigated for opportunities to shift assets to better ensure higher impacting programs are funded first. In fact, if there are other programs not listed in this data set that could provide more benefit than these four, it should be investigated to ensure programs D, E, G, and J are not being acquired at the detriment of programs that could be more valuable.

Next, the Julia coding in this thesis for TOB calculations is 9,000 times faster than the current R model. While it is unreasonable for AFC to evaluate and compare all 2.78 nonillion combinations, at this new speed it is reasonable to conduct a wide array of macro

and micro-level analysis. Therefore, this coding should be used and the criteria established in this thesis in terms of "most" and "least" effective, should be codified and adopted so that future analysis can provide rapid comparisons of TOBs using a metric other than a raw score.

Finally, the machine learning model was over 40 times faster than the Julia coding. While this speed would allow for faster analysis, it is difficult to imagine a half million comparisons being compared during a single analysis period. Additionally, the machine learning model proved to have accuracy problems that are large enough to question the validity of choosing speed over accuracy. For example, is a relative error of 1.3% acceptable when allocating billions of dollars to a mixture of programs that isn't necessarily the best combination? Therefore, the Julia coding initially developed for the TOB should be used instead of the ML model because it follows the current AFC algorithm, and is therefore deemed accurate. However, this initial investigation into a ML model provides an opportunity for future research as the dependency matrix and financial constraints are potentially added to the overall calculations.

6.2 Future Direction

The possibility of dependencies between programs and portfolios is elaborated on in section 2.3.1 with a discussion of the dependency matrix. AFC is currently establishing that matrix and, as more information is gathered or developed, a sensitivity analysis would prove how influential it is on the specific portfolios and programs. Future research could look at trends, outcomes, and sensitivities of a dependency matrix.

The three modernization levels of 0%, 50%, and 100%, as stated in Section 2.3.1, are potentially an oversimplification of the problem. For example, it is possible that a formation of trucks could be modernized to 25%, meaning one of the four trucks is replaced. This would create many more possibilities, but would offer an opportunity to further research the validity of programs. For example, Program D has scores of 24.33, 27.444, 30.555. The initial impression is "why should AFC modernize Program D from 0% to 50% when the return is only 3 in terms of the Capability Value?" These quick identifications may become more prevalent, or recognizable, if there were more than three modernization levels. Future

research could look into the possibilities of more modernization levels, if the data becomes available.

This thesis did not include a financial analysis, but the best way to enhance this analysis is to incorporate costs. For example, a finding of this thesis is that Programs S and X provide nearly the same benefit for Theater A. Future research could look at the question "are these differentiable through finances or procurement to better segregate the impact of the two programs?" Additionally, are there time constraints that aren't considered in our current model?

Finally, many other ML models exist in Julia, R, and Python, as well as opportunities to explore parallel computing, GPU, and MPI approaches. Using these different software programs, and/or hardware combinations, coupled with different levels of computer precision, creates countless possibilities. For example, this thesis used double precision in some areas, and single precision in the ML model. Research into the speed, accuracy, and effectiveness of altering from double precision to single, or half precision, may prove to be of use to AFC when implementing these equations on lowered computing platforms like tablets and phones.

APPENDIX: Analysis of Portfolio A

Ranking	Total Score	Program A	Program B	Program C	Program D
1	100.00	100	100	100	100
2	99.96	100	100	100	50
3	99.92	100	100	100	0
4	99.71	100	100	50	100
5	99.69	50	100	100	100
6	99.68	100	100	50	50
7	99.65	50	100	100	50
8	99.64	100	100	50	0
9	99.61	50	100	100	0
10	99.47	100	50	100	100
11	99.43	100	50	100	50
12	99.40	50	100	50	100
13	99.39	100	50	100	0
14	99.36	50	100	50	50
15	99.33	50	100	50	0
16	99.19	100	50	50	100
17	99.16	50	50	100	100
18	99.15	100	50	50	50
19	99.12	50	50	100	50
20	99.11	100	50	50	0
21	99.08	50	50	100	0
22	99.08	100	100	0	100
23	99.04	100	100	0	50
24	99.00	100	100	0	0
25	98.88	50	50	50	100

Ranking	Total Score	Program A	Program B	Program C	Program D
26	98.84	50	50	50	50
27	98.80	50	50	50	0
28	98.77	50	100	0	100
29	98.73	50	100	0	50
30	98.69	50	100	0	0
31	98.69	0	100	100	100
32	98.65	0	100	100	50
33	98.61	0	100	100	0
34	98.55	100	50	0	100
35	98.51	100	50	0	50
36	98.48	100	50	0	0
37	98.40	0	100	50	100
38	98.36	0	100	50	50
39	98.32	0	100	50	0
40	98.24	50	50	0	100
41	98.20	50	50	0	50
42	98.19	100	0	100	100
43	98.16	50	50	0	0
44	98.16	0	50	100	100
45	98.15	100	0	100	50
46	98.12	0	50	100	50
47	98.11	100	0	100	0
48	98.08	0	50	100	0
49	97.91	100	0	50	100
50	97.88	50	0	100	100

Ranking	Total Score	Program A	Program B	Program C	Program D
51	97.87	0	50	50	100
52	97.87	100	0	50	50
53	97.84	50	0	100	50
54	97.83	0	50	50	50
55	97.83	100	0	50	0
56	97.80	50	0	100	0
57	97.80	0	50	50	0
58	97.77	0	100	0	100
59	97.73	0	100	0	50
60	97.69	0	100	0	0
61	97.59	50	0	50	100
62	97.56	50	0	50	50
63	97.52	50	0	50	0
64	97.27	100	0	0	100
65	97.24	0	50	0	100
66	97.23	100	0	0	50
67	97.20	0	50	0	50
68	97.20	100	0	0	0
69	97.16	0	50	0	0
70	96.96	50	0	0	100
71	96.92	50	0	0	50
72	96.88	50	0	0	0
73	96.88	0	0	100	100
74	96.84	0	0	100	50
75	96.80	0	0	100	0

Ranking	Total Score	Program A	Program B	Program C	Program D
76	96.59	0	0	0	100
77	96.55	0	0	0	50
78	96.51	0	0	50	0
79	95.96	0	0	0	100
80	95.92	0	0	0	50
81	95.88	0	0	0	0

List of References

[1] Department of Defense. "Army Futures Command: Leading the transformational modernization of the U.S. Army." Accessed Apr. 18, 2021. [Online]. Available: https://www.army.mil/futures#org-about

[2] "Interpreted vs compiled programming languages: What's the difference?" FreeCodeCamp, January 20, 2020. [Online]. Available: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

[3] Julia Programming Language. "Mulit-dimensional arrays," May 6, 2021. [Online]. Available: https://docs.julialang.org/en/v1/manual/arrays/

[4] *Doctrine for the Armed Forces of the United States*, ADP 3-0, Headquarters, Department of the Armyf, Washington, DC, USA, 2019. [Online]. Available: https://armypubs.army.mil/epubs/DR_pubs/DR_a/ARN18010-ADP_3-0-000-WEB-2.pdf

[5] J. McClary, private communication, Mar. 2021.

[6] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning, second edition*. MIT Press, 2018. Available: https://books.google.com/books?id=dWB9DwAAQBAJ

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/pdf/1412.6980v5.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California