



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**AN ANALYSIS OF THE MARINE CORPS
DEPOT-LEVEL MAINTENANCE BUDGETING MODEL**

by

Thomas P. Mckavitt III

June 2019

Thesis Advisor:
Second Reader:

Thomas W. Lucas
Matthew T. Geiser

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2019	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE AN ANALYSIS OF THE MARINE CORPS DEPOT-LEVEL MAINTENANCE BUDGETING MODEL			5. FUNDING NUMBERS NPS-19-M278	
6. AUTHOR(S) Thomas P. Mckavitt III				
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) Marine Corps Logistics Command, Albany, GA 31704			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 Marine Corps relies on advanced ground equipment that requires significant depot-level maintenance to extend its life cycle. However, Marine Corps budgets continue to be strained, and every dollar counts. Therefore, the Marine Corps Logistics Command has a vested interest in optimizing its budget to provide the best possible support to the warfighter. The Repair Optimization Materiel Evaluator (ROME) model is used to assist in creating repair plans. Using data-farming software designed specifically for this thesis and advanced design of experiments, the ROME model was run more than 2,000 times with specific, traceable, and repeatable changes made to the inputs each time. The outputs from these runs were analyzed to determine the model's efficacy. Analysis determined that the model's reliance on warfighting values and small penalty terms has created a bias toward allocating funds for high-density, low-cost items with low warfighting values at the expense of items with much higher warfighting values. Specifically, if monies were allocated only according to the ROME model, it would default to fixing thousands of M-16 rifles and rifle optics and very few, if any, large pieces of equipment such as tanks. Based on the findings from this thesis, it is recommended that ROME's objective function be reformulated or calculation of warfighting values be changed. The new data farming capabilities will allow testing these modifications as well as enable advanced analysis with ROME.				
14. SUBJECT TERMS Marine Corps Logistics Command, depot maintenance, LOGCOM, simulation, ROME Model, Marine Corps, logistics, maintenance, data farming, SEED Center			15. NUMBER OF PAGES 113	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**AN ANALYSIS OF THE MARINE CORPS DEPOT-LEVEL MAINTENANCE
BUDGETING MODEL**

Thomas P. Mckavitt III
Captain, United States Marine Corps
BS, U.S. Naval Academy, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
June 2019**

Approved by: Thomas W. Lucas
Advisor

Matthew T. Geiser
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Marine Corps relies on advanced ground equipment that requires significant depot-level maintenance to extend its life cycle. However, Marine Corps budgets continue to be strained, and every dollar counts. Therefore, the Marine Corps Logistics Command has a vested interest in optimizing its budget to provide the best possible support to the warfighter. The Repair Optimization Materiel Evaluator (ROME) model is used to assist in creating repair plans. Using data-farming software designed specifically for this thesis and advanced design of experiments, the ROME model was run more than 2,000 times with specific, traceable, and repeatable changes made to the inputs each time. The outputs from these runs were analyzed to determine the model's efficacy. Analysis determined that the model's reliance on warfighting values and small penalty terms has created a bias toward allocating funds for high-density, low-cost items with low warfighting values at the expense of items with much higher warfighting values. Specifically, if monies were allocated only according to the ROME model, it would default to fixing thousands of M-16 rifles and rifle optics and very few, if any, large pieces of equipment such as tanks. Based on the findings from this thesis, it is recommended that ROME's objective function be reformulated or calculation of warfighting values be changed. The new data farming capabilities will allow testing these modifications as well as enable advanced analysis with ROME.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND	1
B.	OBJECTIVE	2
C.	SCOPE, LIMITATIONS, AND ASSUMPTIONS	3
D.	LITERATURE REVIEW	4
II.	REPAIR OPTIMIZATION MATERIEL EVALUATOR MODEL (ROME)	7
A.	ROME OVERVIEW	7
B.	ROME FORMULATION	7
III.	METHODOLOGY	15
A.	ROME FARMER OVERVIEW	15
B.	DESIGN OF EXPERIMENTS	16
1.	Table of Authorized Materiel Control Number Selection.....	16
2.	Input Variation.....	17
IV.	ANALYSIS	21
A.	INITIAL MARINE CORPS LOGISTICS COMMAND DESIGN OF EXPERIMENTS.....	21
B.	WARFIGHTING RANKS	27
C.	MAXIMIZING UNSERVICEABLE RETURNS BY TABLE OF AUTHORIZED MATERIEL CONTROL NUMBER	33
V.	CONCLUSIONS AND RECOMMENDATIONS.....	42
A.	STUDY CONCLUSIONS.....	42
B.	RECOMMENDATIONS.....	42
1.	Marine Corps Logistics Command Recommendations.....	42
2.	Marine Corps Recommendations.....	44
	APPENDIX A. ROME USER’S MANUAL	45

APPENDIX B. ROMEFARMER USER’S MANUAL	75
LIST OF REFERENCES	89
INITIAL DISTRIBUTION LIST	91

LIST OF FIGURES

Figure 1.	Score Slopes. Source: DeMarco (2014).....	12
Figure 2.	Top Ten Sorted Parameter Estimates of the Stepwise Regression Results from JMP	22
Figure 3.	Mean and Median Objective Function Value by Budget.....	25
Figure 4.	Median Number Repaired by Budget for the Five Highest ICOST Items.....	26
Figure 5.	Median Number Repaired by Budget for the Five Lowest ICOST Items.....	26
Figure 6.	Top Ten Sorted Parameter Estimates for Stepwise Regression with Warfighting Ranks	29
Figure 7.	Objective Value by Budget with Warfighting Ranks	29
Figure 8.	Median Number of Items Repaired of Each of the Top Five Warfighting Value TAMCNs by Budget.....	30
Figure 9.	Median Number of Items Repaired of Each of the Top Five Warfighting Rank TAMCNs by Budget	31
Figure 10.	Median Number of Items Repaired of Each of the Bottom Five Warfighting Value TAMCNs by Budget.....	32
Figure 11.	Median Number of Items Repaired of Each of the Bottom Five Warfighting Rank TAMCNs by Budget	32
Figure 12.	Top Ten Sorted Parameter Estimates of the Stepwise Regression Results using Warfighting Values from JMP	34
Figure 13.	Top Ten Sorted Parameter Estimates of the Stepwise Regression Results using Warfighting Ranks from JMP	34
Figure 14.	Mean and Median Objective Function Value by Budget using Warfighting Value	35

Figure 15.	Mean and Median Objective Function Value by Budget using Warfighting Rank.....	35
Figure 16.	Median Items Repaired for Top Five Warfighting Value TAMCNs.....	36
Figure 17.	Median Items Repaired for Top Five Warfighting Ranks TAMCNs	37
Figure 18.	Median Items Repaired for Bottom Five Warfighting Values TAMCNs.....	38
Figure 19.	Median Items Repaired for Bottom Five Warfighting Ranks TAMCNs.....	38

LIST OF TABLES

Table 1.	TAMCNs Used for Experimentation	17
Table 2.	Pairwise Correlation Table of Ten Input Variables	19
Table 3.	TAMCNs Sorted by Repair Costs.....	22
Table 4.	TAMCNs Sorted by Warfighting Value	24
Table 5.	TAMCNs Sorted by Warfighting Rank	28
Table 6.	TAMCNs Sorted by Warfighting Value per Dollar of Repair Cost	40

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAV	Amphibious assault vehicle
DERO	Dynamic Equipment Repair Optimization
DOE	Design of experiments
DON	Department of the Navy
DRACM	Dimension Reduction and Clustering Model
GAMS	General Algebraic Modeling System
HMMWV	High mobility multipurpose wheeled vehicle
IED	Improvised explosive device
LAV	Light Armored Vehicle
MARCORLOGCOM	Marine Corps Logistics Command
MIP	Mixed integer program
NOB	Nearly orthogonal-and-balanced
NOLH	Nearly orthogonal Latin hypercube
NRFI	Not ready for issue
PEI	Principal end item
POM	Program objectives memorandum
PP&O	Deputy Commandant for Plans, Policies, and Operations
RCO	Rifle combat optic, optic for M-16 family of rifles
RFI	Ready for issue
ROME	Repair Optimization Materiel Evaluator
SAID	Studies, Analysis, and Innovation Division, formerly Studies Analysis Division
SEED	Simulation Experiments and Efficient Designs
SME	Subject matter expert
TAMCN	Table of authorized materiel control number
USR	Unserviceable returns
WMR	War materiel requirement
WSMC	Weapon Systems Management Center

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The United States Marine Corps grows heavier and heavier each year. New equipment, technology, and capabilities enhance combat performance but also strain already lean maintenance budgets. The current Marine Corps' depot-level maintenance budget does not actually cover all of the required annual maintenance. This means the Marine Corps must optimize every dollar allocated to depot-level maintenance.

The Marine Corps Logistics Command (MARCORLOGCOM) has relied on a mixed integer programming (MIP) optimization model, known as the Repair Optimization Materiel Evaluator (ROME), to assist in ensuring effective allocation of depot-level maintenance budgets. Developed by an Operations Analyst in the Studies and Innovation Division (SAID) of MARCORLOGCOM, ROME is programmed to determine how many of each of the 209 different table of authorized materiel control number (TAMCN) items should be repaired each fiscal year. This is done by considering over 15 factors for each TAMCN item, including Marine Corps priorities, current TAMCN readiness, estimated budgets available, and the cost of repairing each item. All of these inputs are balanced against each other, according to the various constraints in the model, to determine the optimal way of allocating the depot-level maintenance budget.

Until this thesis, MARCORLOGCOM had no efficient way of understanding how ROME's outputs would change due to minor or, even, major changes in the inputs. A comprehensive exploration would require each of the over 4,000 individual input values to be changed manually, the model run, and the output analyzed to determine what was affected. This is obviously a prohibitively time-consuming process if more than one input variable needs to be changed across a range of values. As a part of this thesis research, the ROMEfarmer was created. ROMEfarmer is a sophisticated software wrapper that modifies the ROME model to accept and run an advanced design of experiments to execute a predetermined amount of model runs, automatically. The outputs from these runs are combined into a single output document for analysis. ROMEfarmer can execute specific, traceable, and repeatable changes to the input variables that allow the end user to

understand how changes in budgets, equipment numbers, or repair costs can alter budget allocations.

With the development of ROMEfarmer advanced analysis of ROME can now be accomplished in a fraction of the time. Using advanced design of experiments (DOE) methods created at the Simulation Experiments and Efficient Designs (SEED) Center at Naval Postgraduate School (NPS), a thorough examination and sensitivity analysis of ROME was performed. The DOE used in this thesis required the ability to examine both continuous and discrete values for several input variables. This was accomplished using a nearly orthogonal and balanced (NOB) DOE. The NOB used in this thesis had 512 design points consisting of 46 different input variables. Each time ROMEfarmer was run it was as if ROME was run 512 times with a different mix of input values. The design was so efficient that ROMEfarmer could complete these runs in under 10 minutes.

Working with MARCORLOGCOM, the scope of the data that ROME works with was scaled down for ease of analysis and to focus on specific TAMCN items. Of the 209 TAMCNs that ROME optimizes, 15 were chosen for analysis. These 15 TAMCNs represent a good spread of high-density, low-cost items such as rifles, and low-density, high-cost items such as main battle tanks. Each of these 15 TAMCNs had three input variables altered through the DOE. Those inputs, combined with altering the depot-level maintenance budget, account for the 46 inputs that made up the DOE. Over the four different experiments that were run for this thesis, more than two thousand individual runs of ROME were executed.

The first experiment was a baseline run where each of the 15 TAMCNs had their readiness percentage goal altered from 65% to 95% in 5% intervals, their number of repaired items ready to ship changed from the fiscal year 2020 baseline by +/- 10%, and the cost of repairing the individual items changed from the same baseline by +/- 10%. The budget was changed from a baseline of \$285 million by +/- 10% in \$5 million intervals. The initial results were quite interesting, demonstrating a clear bias in the model toward repairing high-density, low-cost items to the detriment of low-density, high-cost items.

The second experiment sought to counter this bias by replacing an input variable, known as warfighting value, with a warfighting rank. The warfighting value is a value assigned to each TAMCN item that prioritizes that item against all others in the ROME model. Currently, the warfighting values are numbers; between 1.27 for the E0017A4 Rifle Combat Optic (RCO) and 8.06 for a Husky mine detection vehicle. This means that there is not enough differentiation in warfighting values between the various TAMCNs to substantially alter the model's output. To address this, ROMEfarmer was run again substituting a simple ranking from one to 209, with 209 being the highest value TAMCN, to determine if a rank would provide enough differentiation for the various TAMCNs. This did result in some improvement. The number of repaired items for the lower ranked TAMCNs dropped significantly and more low-density, high-cost TAMCNs were repaired. However, not as much as was expected.

In consultation with MARCORLOGCOM the final two experiments were designed to see what ROME would repair if everything in the Marine Corps needed depot-level maintenance at the same time. To create this scenario, the unserviceable returns, number of broken equipment by TAMCN, were set equal to the required number of items needed in service. With this adjustment, ROMEfarmer was run using both the warfighting values and warfighting ranks. Both experiments resulted in only high-density, low-cost items, such as rifles and rifle scopes being repaired in significant numbers. This starkly demonstrated ROME's bias towards repairing these items.

The conclusions from this research point to a need to reformulate the objective function in the ROME model. The current function does not sufficiently prioritize low-density, high-cost items. This is likely due to the use of a small penalty term that is easily overcome by the number of high-density, low-cost items that can be repaired for a fraction of the cost of the more valued, but more expensive, low-density items. It is recommended that in order for MARCORLOGCOM to improve the usefulness of ROME results, another model, the Dimension Reduction and Clustering Model (DRACM), that creates the warfighting values that feed into ROME, should also be thoroughly examined with an eye toward improving the utility of its output. Additionally, the warfighting values that are in current use prioritize equipment that does not appear to meet the future

needs of the Marine Corps. For example, the highest valued item is the B0063 Husky, a mine detection vehicle, that was designed to find improvised explosive devices in desert terrain. As the Marine Corps prepares for great power competition this item will have much less value that it did during the conflicts in Iraq and Afghanistan.

The development of ROMEfarmer, in conjunction with the findings of this thesis, demonstrate the shortcomings of ROME. To improve the optimization of the Marine Corps' depot-level maintenance budget, further study of, and improvement to, ROME is required. The recommended changes in this thesis have the potential to greatly improve the depot-level maintenance budgeting process to the benefit of the warfighter on the ground.

ACKNOWLEDGMENTS

First, I must acknowledge and thank my wife, Emily, for acting as a single parent during my last quarter at NPS to ensure I had the time and attention to devote to finishing this thesis and my degree. I must also thank my advisor, Professor Tom Lucas, for his guidance throughout this entire project. His scheduling, revisions, and connections to MARCORLOGCOM brought everything together for this project. My second reader, CDR Matthew Geiser, assisted me with timely edits and advice from a senior officer perspective, which made this thesis better. To the captains at MARCORLOGCOM, Tony Vanderzee and Travis Neesmith, thanks for making me feel welcome on my visits and providing me insights on the use of ROME and fielding my many questions about depot maintenance.

Finally, I must acknowledge the great support I received from the SEED Center research associates, Steve Upton and Mary McDonald. Steve created ROMEfarmer and executed all the runs I needed to create data for this thesis, usually within a day of me submitting my DOE. Mary was my JMP guru and helped me immensely in the analysis of the outputs. She formatted the ROMEfarmer outputs in JMP, which allowed me to focus on analysis and writing. I am forever grateful to them both for all their hard work and cheerful assistance.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The United States Marine Corps, as a component of the Department of the Navy (DON), is required to compete with the United States Navy for total allocated DON monies. This has led to a lean fighting force that is inclined to do more with less. However, it also requires the Marine Corps to attempt to maximize the return on each dollar allocated to it. This includes the critical capability of depot-level maintenance and repair.

Depot-level maintenance is defined by Title 10 of the United States Code as “materiel maintenance or repair requiring the overhaul, upgrading, or rebuilding of parts, assemblies, or subassemblies” (United States Code 2019). This includes taking in Marine Corps ground equipment, everything from rifles to tanks, repairing and refurbishing them, and then returning them to the operating forces for continued use. The readiness of Marine Corps fighting units depends on this. The maintenance is carried out at the Marine Corps logistics bases in Albany, Georgia and Barstow, California. This effort is overseen by the Marine Corps Logistics Command (MARCORLOGCOM). This process is extremely complex because it must balance the equipment needs of the active forces, reserves, maritime prepositioned forces, and required war reserves (equipment activated in time of large-scale conflict).

According to the directives that govern depot-level maintenance, including the Marine Corps Technical Publication 3–40E Maintenance Operations, every piece of ground equipment that is eligible for depot-level maintenance is cyclically scheduled for induction into one of the maintenance depots to attempt to repair before replacement is required (Department of the Navy [DON] 2016). The publication further requires that equipment that becomes significantly damaged or degraded beyond the unit’s organic capability will also be sent for depot-level repair. Each different type of equipment has a unique table of authorized materiel control number, or TAMCN, which is how the equipment is tracked when allocated funding.

All of these competing interests must be balanced to find an optimal way to spend allocated funds, funds that are steadily decreasing year to year. In the fiscal year 2019 budget, the Marine Corps requested \$334 million for depot-level maintenance and repair; however, this would only fund 80% of the executable requirements (Department of Defense [DoD] 2018). All of this is to emphasize that while the importance of depot-level maintenance and repair has not changed, funding availability has decreased. In order to ensure that the fewer depot-level maintenance dollars are efficiently used, MARCORLOGCOM created the Repair Optimization Materiel Evaluator (ROME) model. This model's purpose is to optimize the distribution of allocated money to ensure the Marine Corps repairs the proper mix of equipment (DeMarco 2014). Ultimately, the implementation of this model assists in guaranteeing the nation a lethal and ready fighting force. The actual composition of the model, including inputs and outputs, is examined in Chapter II of this thesis.

B. OBJECTIVE

The objective of this thesis is to examine the current ROME model to determine how robust the output of the model is to changes in its inputs. This will ultimately lead to recommendations on how to improve the model to provide output that is more informative to end users at MARCORLOGCOM. This thesis is guided by the following questions:

- Can data farming methods identify robust ROME solutions?
- What inputs are ROME's outputs most sensitive to?
- Are there new design of experiment methods and/or other MARCORLOGCOM generated inputs that can be used to enhance the ROME model?

Recommendations from this thesis will guide users of ROME and follow-on research in an attempt to continually improve ROME. This, in turn, will ensure the

Marine Corps has an optimal solution when balancing the various competing priorities that depot-level maintenance must consider in enabling a lethal, ready, and capable force.

C. SCOPE, LIMITATIONS, AND ASSUMPTIONS

(1) Scope

This thesis focuses on testing the robustness of the ROME model and conducting a sensitivity analysis on the outputs to determine which of the inputs effects the model the most. Further, the research improves upon the ability of analysts to experiment with ROME and create alternate outputs in the event of dynamic changes to allocated resources such as budget and available equipment.

(2) Limitations

There are two limitations to this thesis that deserve to be highlighted. First, some of the inputs have been classified since this research began; therefore historic data is used in lieu of current, classified data to ensure this research is readily accessible for future work and all end users of ROME. Additionally, due to the methodology of the thesis, which is described in detail in Chapter III, not all of the individual types of equipment can be tested using variability in their inputs. Therefore, working with MARCORLOGCOM, specifically the Weapons Systems Management Center (WSMC), a specific group of items with various benefits to the research were selected for experimentation, this is described in detail in Chapter III. This same limitation also caused the analysis to focus on only one fiscal year despite the model's ability to provide output for six fiscal years.

(3) Assumptions

The main assumption in this thesis relies on the data used to initiate the experimentation. That is, it is assumed that the data used as the baseline is accurate and reflects a typical running of the model. By making this assumption, the experimentation can be grounded on this baseline data and the results treated as typical for the purpose of

sensitivity analysis. Additional assumptions lie in the model itself, which is further examined in Chapter II.

D. LITERATURE REVIEW

Literature concerning optimization of processes in the Marine Corps, and the greater Department of Defense, abounds. However, for this thesis, emphasis is placed on literature pertaining to depot-level maintenance scheduling and overall optimization of maintenance operations within the Marine Corps. While this limits the available literature, it also serves to focus the reader and researcher on the most pertinent topics.

Work on improving Marine Corps depot-level maintenance using optimization dates back to at least 1995. An integer linear program model was created that optimized depot-level maintenance for tanks (Barger 1995). However, optimizing for one single TAMCN excludes the thousands of others that are competing for maintenance funds. A similar approach was used to attempt to improve an existing model by focusing solely on the light armored vehicle (LAV) (Mullins et al. 2005).

The first large scale optimization model for Marine Corps depot-level maintenance was created by Goodhart while working for the Marine Corps' Deputy Commandant for Installations and Logistics (Goodhart 1999). Goodhart created a linear programming optimization model that focused on returning the highest possible readiness rate for all TAMCNs eligible for depot-level maintenance. His model was named the Dynamic Equipment Repair Optimization (DERO). DERO attempted to model the many inputs into the real-world decision-making process that, to that point, had consisted of subject matter experts (SMEs) ranking each different TAMCN and doing calculations for budget allocations to each by hand. The result of DERO was a significant increase in readiness for various TAMCNs across the Marine Corps (Goodhart 1999).

Once DERO was used routinely, data could be collected concerning its ability to accurately optimize the depot-level maintenance budget. It quickly became clear that DERO was appropriate for a static optimization problem. However, there is nothing static about budgets and numbers of pieces of equipment requiring repair at any given time.

Therefore, the model needed to be run and re-run each time there was even a slight change. It was noticed that when the inputs were changed, even slightly, a significantly different resulting list of TAMCNs to be repaired was returned (Drexler 2003). This led to an effort to incorporate persistence into the model. Previous work with persistence in optimization had shown that to incorporate persistence a baseline run of the model was needed; then, all subsequent runs of the model would penalize large deviations from the baseline (Brown et al. 1997). The new penalty term incorporated by Drexler allowed for subsequent runs of the model with only minor changes. This is critically important because a new run of the model may be needed even after the initial model solution has begun to be carried out.

As work on DERO progressed, it was essential for non-optimizers to approach improving the model from a different perspective. This is what Mullins and others attempted to do by looking specifically at the LAV. An attempt was made to examine every aspect of a single TAMCN's process through the depot-level maintenance program. The results were focused on improving the process and did not focus on optimization or any modeling (Mullins et al. 2005). However, Mullins did highlight issues with repair cycle time and repair cost estimate variability. All of this work came together in the development of the ROME model.

DeMarco sought to combine the best of all previous research in building off of Goodhart's work on DERO and Drexler's work with persistence (DeMarco 2014). This resulted in the ROME model, on which this thesis is centered.

THIS PAGE INTENTIONALLY LEFT BLANK

II. REPAIR OPTIMIZATION MATERIEL EVALUATOR MODEL (ROME)

A. ROME OVERVIEW

The ROME model is used to assist in the distribution of allocated depot-level maintenance funds across the many TAMCNs that qualify for depot-level maintenance. The model is used to inform MARCORLOGCOM's Program Objectives Memorandum (POM) process. The POM is how MARCORLOGCOM justifies its budget requests to Marine Corps Headquarters (Coleman 2018). ROME assists this process by optimizing the distribution of the next five predicted fiscal year depot-level maintenance budgets. The ROME output demonstrates how MARCORLOGCOM will allocate money requested for depot-level maintenance. Further, because the model deals with individual TAMCN items, ROME is a useful tool for Marine Corps program managers and commodity managers in predicting the readiness of their respective equipment programs.

The model is a mixed integer linear program that takes in twenty-two input documents. The majority of the input documents contain a value for each TAMCN for each fiscal year, others provide the budget by fiscal year or binary arguments. ROME provides seventeen outputs that outline which, and how many of each, TAMCNs are budgeted for repair in the next five fiscal years. The inputs and outputs are described in detail in the ROME formulation section.

B. ROME FORMULATION

The ROME model is formulated to maximize the objective function value. The objective function takes into consideration warfighting values, readiness goals, and the current readiness of each TAMCN in each fiscal year. The higher the objective function value, the better Marine Corps' readiness will be. A negative objective function value means that the model failed to repair all of the TAMCN items that required repair during that fiscal year. The negative value comes from the use of penalty terms, described below, that penalize the objective function value for failing to meet repair goals.

Execution of the ROME model is a complex process that requires culling and cleaning data from several sources located at MARCORLOGCOM. The data must then be formatted correctly for ROME to read it in properly. In order to ease this complicated process, the author of the model created a step-by-step guide entitled Repair Optimization Materiel Evaluator (ROME) User's Manual (DeMarco 2014). This manual is referred to as the ROME User's Manual in this thesis.

All of the following formulations, and most explanations, come directly from the ROME User's Manual. In some cases, extra context is added by the author to increase the comprehension for non-MARCORLOGCOM readers. The full ROME User's Manual is included as Appendix A to put the supporting documentation of ROME and ROMEfarmer (see Appendix B) in one document. ROMEfarmer is the software developed for this thesis that enables us data farm ROME (see Chapter III) and the user's manual is included in Appendix B.

Indices:

$s \in S$ (e.g. $S = \{1, 2, 3, 4, 5\}$): indices for the lines which bound the $score_{t,f,y}$ variable used in the objective function

$t \in T$ (e.g. $T = \{A0004, A0017\}$): TAMCNs of interest

$f \in F$ (e.g. $F = \{ACTIVE, RESERVES\}$): units of interest

$y \in Y$ (e.g. $Y = \{2019, 2020, \dots, 2023\}$): years of interest

Parameters:

$wmr_{t,f,y}$: war materiel requirement for t from f during y

$usr_{t,f,y}$: unserviceable returns, PEIs that are sent from operating units for depot-level maintenance, that are forecast to occur of t from f during y

$wo_{t,f,y}$: washouts, items inducted into depot maintenance but are too broken to be worth repairing, that are forecast to occur of t from f during y

$rtgt_{t,f,y}$: readiness target, a percentage of total TAMCN readiness, of t for f during y
 $icost_{t,y}$: repair cost per unit for t during y
 $issue_{t,y}$: number of t forecast to be issued during y from new procurement
 $budget_{f,y}$: the budget for f during y for depot-level maintenance
 $srft_{t,f}$: starting number of stratified ready for issue (RFI) t at f in year 0
 $sbl_{t,f}$: starting number of stratified not ready for issue (NRFI) t at f in year 0
 $uspare0_t$: number of RFI t not stratified (excess) in year 0
 $dspare0_t$: number of NRFI t not stratified (excess) in year 0
 $unfF0_{t,f}$: number of number of condition code F (in need of depot-level repair) t at f in year 0
 $value_t$: warfighting value of t , allows for subjectively placing more or less value on each TAMCN. Recommended value is determined using the Dimension Reduction and Clustering Model (DRACM), which is then certified, and classified, by Headquarters, Marine Corps (Coleman 2018).
 $slope_s$: slope of line s used in bounding the $score_{t,f,y}$ variable
 $intercept_s$: intercept of line s used in bounding the $score_{t,f,y}$ variable
 $discount$: fractional value of repairing an item in year y instead of year $y - 1$
 $limRpr$: binary set to 1 if rpr is limited from above to usr
 $fixFund$: binary set to 1 if a funding minimum is to be set
 $minFund$: the fractional amount of the maximum funding to use for the minimum
 y_0 : the first year in Y

$\text{tilim}_{t,f,y}$: the maximum allowed number of t which may become excess from f during y .

This parameter is used in the logic which regulates the transfer of items to and from excess. It is calculated as follows:

$$\text{tilim}_{t,f,y} = \max\{0, \text{wmr}_{t,f,y-1} - \text{wmr}_{t,f,y}\} \quad \forall t, f, y \mid y > y_0$$

$$\text{tilim}_{t,f,y_0} = \max\{0, \text{srfi}_{t,f} + \text{sbl}_{t,f} - \text{wmr}_{t,f,y_0}\} \quad \forall t, f$$

pen : a penalty for being short of $\text{rtgt}_{t,f,y}$, for using $\text{cheat}_{t,f}$ variable or for being in excess of $\text{wmr}_{t,f,y}$

$$\text{pen} = 2 \times \max_t \{\text{value}_t\}$$

Variables:

$\text{score}_{t,f,y}$: a measure of the readiness of t in f during y

$\text{rpr}_{t,f,y}$: the number of t in f at the end of y

$\text{rfi}_{t,f,y}$: the number of RFI, items that have been repaired at a depot but not yet sent to a using unit, t in f at the end of y

$\text{nrfi}_{t,f,y}$: the number of NRFI, items still in depot maintenance, t in f at the end of y

$\text{isrfi}_{t,y}$: excess RFI t at the end of y

$\text{isnrfi}_{t,y}$: the number of excess NRFI t at the end of y

$\text{unfF}_{t,f,y}$: the number of condition code F, items identified as in need of depot maintenance but have not been inducted into depot maintenance yet, t in f at the end of y

$\text{tist}_{t,f,y}$: the number of RFI t which became excess in f during y

$\text{tiu}_{t,f,y}$: the number of NRFI t which became excess in f during y

$\text{float}_{t,f,y}$: excess RFI t redistributed to f during y

$\text{strn}_{t,f,y}$: excess NRFI t redistributed to f during y

$tedef_{t,f,y}$: the number of t below WMR in f at the end of y

$defind_{t,f,y}$: a binary variable set to 1 if $tedef_{t,f,y} > 0$

$cheat_{t,f}$: variable to allow feasibility in case of poor forecasting, these represent t which had to be stratified to f in the beginning to meet the unserviceable returns forecast

$short_{t,f,y}$: the number of t below the readiness target for f at the end of y

$excess_{t,f,y}$: the number of t above the readiness target for f at the end of y

$aboveWmr_{t,f,y}$: the number of t above the WMR target for f at the end of y

Objective Function:

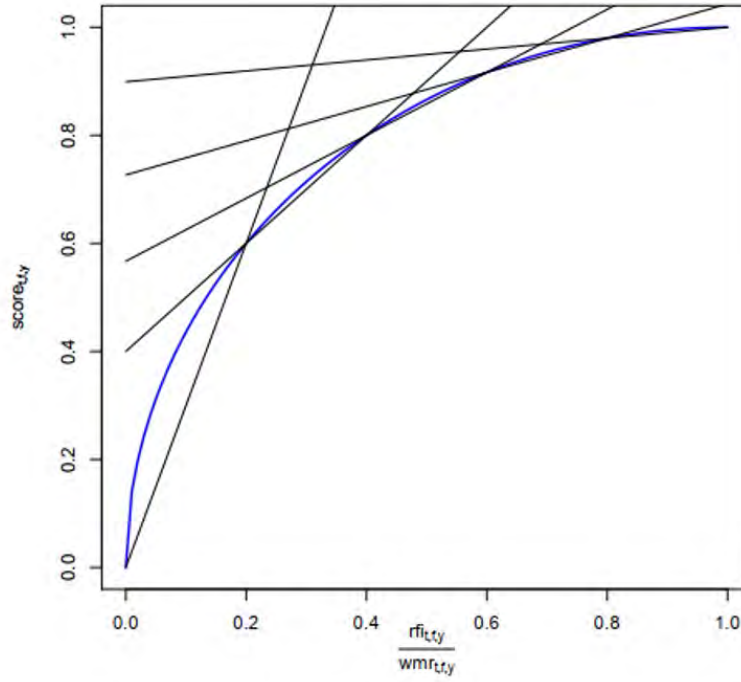
$$\max \sum_{t,f,y} \text{value}_t \times \text{discount}^{y-y_0} \times \text{score}_{t,f,y} - \text{pen} \times \text{short}_{t,f,y} - 2 \times \text{pen} \times \text{cheat}_{t,f} - \text{pen} \times \text{aboveWmr}_{t,f,y}$$

Constraints:

$$\text{Budget constraint: } \sum_t \text{icost}_{t,y} \times \text{rpr}_{t,f,y} \leq \text{budget}_{f,y} \quad \forall f, y$$

$$\text{Minimum funding: } \sum_t \text{icost}_{t,y} \times \text{rpr}_{t,f,y} \geq \text{budget}_{f,y} \times \text{minFund} \times \text{fixFund} \quad \forall f, y$$

$$\text{Score: } \text{score}_{t,f,y} \leq \begin{cases} \text{intercept}_s + \text{slope}_s \frac{\text{rfi}_{t,f,y}}{\text{wmr}_{t,f,y}} \times \text{wmr}_{t,f,y} & \text{wmr}_{t,f,y} > 0 \\ 0 & \text{wmr}_{t,f,y} = 0 \end{cases} \quad \forall s, t, f, y$$



The lines (which approximate the curve shown in blue) bound the $score_{t,f,y}$ variable from above. The effect is that a marginal increase in readiness will lead to a greater marginal increase in $score_{t,f,y}$ when readiness is lower.

Figure 1. Score Slopes. Source: DeMarco (2014)

$$\text{Readiness goal: } rfi_{t,f,y} = rtgt_{t,f,y} - short_{t,f,y} + excess_{t,f,y} \quad \forall t, f, y$$

$$\text{WMR balance: } rfi_{t,f,y} + nrfi_{t,f,y} = wmr_{t,f,y} - tedef_{t,f,y} + aboveWmr_{t,f,y} \quad \forall t, f, y$$

$$\text{Deficiencies: } tedef_{t,f,y} \leq \text{defind}_{t,f,y} \times wmr_{t,f,y} \quad \forall t, f, y$$

$$tedef_{t,f,y} \geq \text{defind}_{t,f,y} \quad \forall t, f, y$$

$$\text{Excess upper bound: } tiu_{t,f,y} + tis_{t,f,y} \leq \text{tilim}_{t,f,y}(1 - \text{defind}_{t,f,y}) \quad \forall t, f, y$$

Excess RFI Balance:

$$isrfi_{t,y} = isrfi_{t,y-1} + \text{issue}_{t,y} + \sum_f (tis_{t,f,y} - \text{float}_{t,f,y}) \quad \forall t, y \mid y > y_0$$

$$\text{isrfi}_{t,y_0} = \text{uspare0}_t + \text{issue}_{t,y_0} + \sum_f (\text{tis}_{t,f,y_0} - \text{float}_{t,f,y_0}) \quad \forall t$$

Excess NRFI balance:

$$\text{insrfi}_{t,y} = \text{isnrfi}_{t,y-1} + \sum_f (\text{tiu}_{t,f,y} - \text{strn}_{t,f,y}) \quad \forall t, y | y > y_0$$

$$\text{insrfi}_{t,y_0} = \text{dspare0}_t + \sum_f (\text{tiu}_{t,f,y_0} - \text{strn}_{t,f,y_0}) \quad \forall t$$

Balance non-excess RFI assets:

$$\text{rfi}_{t,f,y} = \text{rfi}_{t,f,y-1} + \text{rpr}_{t,f,y} + \text{float}_{t,f,y} - \text{tis}_{t,f,y} - \text{usr}_{t,f,y} - \text{wo}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{rfi}_{t,f,y_0} = \text{srfi}_{t,f} + \text{cheat}_{t,f} + \text{rpr}_{t,f,y_0} + \text{float}_{t,f,y_0} - \text{tis}_{t,f,y_0} - \text{usr}_{t,f,y_0} - \text{wo}_{t,f,y_0} \quad \forall t, f$$

Balance non-excess NRFI assets:

$$\text{nrfi}_{t,f,y} = \text{nrfi}_{t,f,y-1} - \text{rpr}_{t,f,y} - \text{tiu}_{t,f,y} + \text{strn}_{t,f,y} + \text{usr}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{nrfi}_{t,f,y_0} = \text{sbl}_{t,f} - \text{rpr}_{t,f,y_0} - \text{tiu}_{t,f,y_0} + \text{strn}_{t,f,y_0} + \text{usr}_{t,f,y_0} \quad \forall t, f$$

Balance condition code F assets:

$$\text{unfF}_{t,f,y} = \text{unfF}_{t,f,y-1} + \text{usr}_{t,f,y} - \text{rpr}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{unfF}_{t,f,y_0} = \text{unfF0}_{t,f} + \text{usr}_{t,f,y_0} - \text{rpr}_{t,f,y_0} \quad \forall t, f$$

Repair limit:

$$\text{rpr}_{t,f,y} \leq \text{usr}_{t,f,y} + (1 - \text{limRpr})\text{unfF}_{t,f,y-1} \quad \forall t, f, y | y > y_0$$

$$\text{rpr}_{t,f,y_0} \leq \text{usr}_{t,f,y_0} + (1 - \text{limRpr})\text{unfF0}_{t,f} \quad \forall t, f$$

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

The methodology of experimentation during this research can be broken into two phases; preparation of the ROME model for experimentation and development of the designs of the various experiments. Both phases were executed in close coordination with MARCORLOGCOM and the Simulation Experiments and Efficient Designs (SEED) Center at Naval Postgraduate School.

A. ROMEFARMER OVERVIEW

In order to prepare the ROME model for experimentation it had to be adapted to accept and execute a design of experiments (DOE). A design of experiments is a matrix where rows represent experiments and columns represent input variable settings for each experiment. Each row is also known as a design point—that is, a unique combination of input variables. Thus, ROME is run over multiple design points, producing data for analysis. To be useful, this process must be automated. This was accomplished through the development of ROMEfarmer by Stephen C. Upton (2018), a SEED Center research associate. Upton’s ROMEfarmer is a sophisticated software wrapper that allows the ROME model to iterate repeatedly until all the design points have been executed. Further, ROMEfarmer provides a usable output for analysis. ROMEfarmer is built on a process known as data farming. Data farming is “purposeful data generation from simulation models” (Lucas et al. 2015, p. 297). Using this process allows for generation of data from specific inputs to an existing model, thus reducing the need to clean output data. This process is specifically explained for this model in the ROMEfarmer User’s Manual:

ROMEfarmer consists of three components: ROMEfarmer, ROMErunner, and ROMEminer. ROMEfarmer is the top-level interface to the user and is responsible for executing the designed experiment. ROMErunner is the component that takes one set of inputs, also called a design point, case or excursion, from a designed experiment and runs the ROME model for that instance. ROMEminer takes the output from the designed experiment and post-processes that output by concatenating the output files into a single output file for further analysis. (Upton 2018, p. 1)

Further information on ROMEfarmer can be found by reading the ROMEfarmer User's Manual found in Appendix B.

B. DESIGN OF EXPERIMENTS

In order to ensure a thorough sensitivity analysis of the ROME model an intelligent DOE was needed. While ROMEfarmer has the capability of reading in an extremely large and sophisticated DOE, time and financial constraints required a focused DOE. MARCORLOGCOM provided guidance for the DOEs in this thesis.

1. Table of Authorized Materiel Control Number Selection

Of the 207 TAMCNs that are eligible for depot-level maintenance, 15 were chosen, these are shown in Table 1. These 15 TAMCNs run the gamut from critical, low-density, high-cost items, such as Abrams main battle tanks, to high-density, low-cost items, such as the rifle combat optic, a scope for the M-16 family of rifles. For this thesis, high-density refers to items that are owned by the Marine Corps in quantities greater than 1,000. Additionally, low-cost refers to items that can be repaired for less than \$100,000 an item.

Table 1. TAMCNs Used for Experimentation

TAMCN	Nomenclature
A15037G*	Air Search RADAR Set (AN/TPS-59A v3)
A21797G*	Microwave Radio Terminal (AN/TRC-170)
B00637B*	Ruggedized Heavy Forklift (TRAM)
B00937B*	Husky (Vehicle Mounted IED Detection System)
B01607B*	Assault Breacher Vehicle
B05897B*	Armored Combat Earthmover
D00037K	Armored Medium Tactical Vehicle Replacement
D00337K	Armored HMMWV
D00527K*	Armored Logistics Vehicle System Replacement
E0017A4	Rifle Combat Optic
E06717M*	Lightweight 155mm Howitzer
E08467K*	Amphibious Assault Vehicle
E09427M*	Light Armored Vehicle, Anti-Tank
E14422M	M-16A4 Combat Rifle
E18887M*	Abrams Main Battle Tank

* Denotes mission essential equipment as defined by the Marine Corps (DON 2018).

Further, these selected TAMCNs represent a sample of all commodity areas for which MARCORLOGCOM is responsible for providing depot-level maintenance. The commodity is denoted by the first letter in the TAMCN, specifically; A: communication-electronics, B: engineer equipment, D: motor transport equipment, and E: weapon systems. The inclusion of each TAMCN was vetted through MARCORLOGCOM, several were included to augment current MARCORLOGCOM initiatives concerning specific items.

2. Input Variation

In order to conduct a thorough sensitivity analysis of the ROME model, controlled randomization of selected input values was required. This began with an examination of the various inputs into the model and the values they assume. A discussion with MARCORLOGCOM led to the selection of three inputs that were varied across the 15 TAMCNs discussed above along with the budget value. The inputs selected were $srfi_{t,f,y}$, $rtgt_{t,f,y}$, $icost_{t,y}$, and $budget_{f,y}$. This resulted in 46 input variables. The inputs

$\text{srfi}_{t,f,y}$, $\text{rtgt}_{t,f,y}$, and $\text{icost}_{t,y}$ were varied by $\pm 10\%$ of the baseline value assigned in fiscal year 2020. The budget was varied by \$5 million increments from \$20 million less to \$30 million more than the fiscal year 2020 budget. Because these factors included both continuous and discrete variable types a mixed design of experiments was required.

With 46 input variables, a brute-force full factorial design of experiment is infeasible. If we assume each input variable has only three levels, then $3^{46} = 8.9 \times 10^{21}$ runs are required to assess all input combinations. Fortunately, Vieira Jr et al. proposed an efficient, nearly orthogonal-and-balanced (NOB) mixed design that allows for the space-filling properties of a nearly-orthogonal Latin hypercube (NOLH) while able to handle both discrete and continuous factor types (Vieira et al. 2013). Using a NOB design tool supplied by the SEED Center (<http://harvest.nps.edu>), it was possible to create a DOE with 512 design points, each of which would become a run of the ROME model with those inputs. Note: as a mixed-integer optimization program, ROME is deterministic; thus, only one run at each design point is required. This initial DOE provided insights that led to follow-on experimentation.

Using the NOB design ensured that while a great deal of input variables are altered over a range of values, pairwise correlation remained low. Table 2 is a pairwise correlation table of ten of the 46 input variables. Only ten were selected to ensure the table was readable. It shows the $\text{srfi}_{t,f,y}$, $\text{rtgt}_{t,f,y}$, and $\text{icost}_{t,y}$ altered for three TAMCNs (chosen alphabetically) with the budget variable. The correlations between these various inputs is negligible, which implies little, to no, confounding of the main effects (see Hernandez et al. [2012]).

Table 2. Pairwise Correlation Table of Ten Input Variables

Correlations										
	RPCT A1503	RPCT A2179	RPCT B0063	BUDGET	ICOST A1503	ICOST A2179	ICOST B0063	SRFI A1503	SRFI A2179	SRFI B0063
RPCT A1503	1.0000	-0.0027	0.0012	-0.0026	0.0066	-0.0040	0.0028	-0.0132	-0.0057	-0.0008
RPCT A2179	-0.0027	1.0000	0.0040	-0.0053	-0.0069	-0.0035	0.0030	0.0154	-0.0020	-0.0028
RPCT B0063	0.0012	0.0040	1.0000	0.0034	-0.0024	0.0086	0.0020	-0.0096	-0.0006	-0.0061
BUDGET	-0.0026	-0.0053	0.0034	1.0000	0.0059	0.0010	0.0022	0.0215	-0.0013	-0.0042
ICOST A1503	0.0066	-0.0069	-0.0024	0.0059	1.0000	-0.0038	-0.0009	0.0044	-0.0033	-0.0002
ICOST A2179	-0.0040	-0.0035	0.0086	0.0010	-0.0038	1.0000	-0.0017	-0.0022	-0.0029	-0.0001
ICOST B0063	0.0028	0.0030	0.0020	0.0022	-0.0009	-0.0017	1.0000	-0.0057	0.0015	-0.0027
SRFI A1503	-0.0132	0.0154	-0.0096	0.0215	0.0044	-0.0022	-0.0057	1.0000	-0.0202	-0.0256
SRFI A2179	-0.0057	-0.0020	-0.0006	-0.0013	-0.0033	-0.0029	0.0015	-0.0202	1.0000	-0.0074
SRFI B0063	-0.0008	-0.0028	-0.0061	-0.0042	-0.0002	-0.0001	-0.0027	-0.0256	-0.0074	1.0000

The correlations are estimated by Row-wise method.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ANALYSIS

A. INITIAL MARINE CORPS LOGISTICS COMMAND DESIGN OF EXPERIMENTS

As described in Chapter III, an initial DOE was created using input from MARCORLOGCOM. Three inputs ($srft_{t,f,y}$, $rtgt_{t,f,y}$, and $icost_{t,y}$) were varied +/- 10% across 15 TAMCNs. Additionally, the fiscal year 2020 budget was varied by \$55 million in \$5 million steps from \$30 million less to \$25 million more than the baseline. The DOE was run in ROMEfarmer by Stephen Upton and the output was organized and pared down by Mary McDonald. This allowed ease in analysis by including only those inputs varied by the DOE and all outputs in a single file.

Analysis of the first output began with a large stepwise regression to estimate the objective function value. The regression included up to all two-way interactions and second order quadratics for all of the 46 inputs in the DOE. This regression identified the inputs and interactions that most impacted the change in the objective function value. Before analysis was begun, it was assumed that the largest driver of change to the objective function value would be the budget. However, the stepwise regression fit, totaling 80 terms, with an adjusted R-square of 0.949, excluded budget as a predictor. Instead, the model fit is dominated by the readiness goal and SRFI inputs for the E0017A4 Rifle Combat Optic (RCO) and the E14422M M-16 rifle and the interactions and quadratics of those terms. As shown in Figure 2, the ten most influential parameters found in the stepwise regression are all related to the SRFI and RPCT for the RCO and M-16.

Sorted Parameter Estimates					
Term	Estimate	Std Error	t Ratio		Prob> t
(RPCT/E0017A4.ALL.2020-0.84951)*(SRFI/E0017A4.ALL-206801)	71.303161	0.400883	177.87		<.0001*
RPCT/E0017A4.ALL.2020	-802896.7	4536.104	-177.0		<.0001*
SRFI/E0017A4.ALL	4.322677	0.027283	158.44		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(RPCT/E0017A4.ALL.2020-0.84951)	-7661828	80316.72	-95.40		<.0001*
(RPCT/E14422M.ALL.2020-0.84971)*(SRFI/E14422MALL-115905)	63.017941	0.701217	89.87		<.0001*
RPCT/E14422M.ALL.2020	-369321	4620.697	-79.93		<.0001*
SRFI/E14422M.ALL	3.3453754	0.05012	66.75		<.0001*
(SRFI/E0017A4.ALL-206801)*(SRFI/E0017A4.ALL-206801)	-0.000139	2.72e-6	-51.12		<.0001*
(RPCT/E14422M.ALL.2020-0.84971)*(RPCT/E14422MALL.2020-0.84971)	-4029180	81384.49	-49.51		<.0001*
(SRFI/E14422M.ALL-115905)*(SRFI/E14422MALL-115905)	-0.000258	8.381e-6	-30.81		<.0001*

Figure 2. Top Ten Sorted Parameter Estimates of the Stepwise Regression Results from JMP

These results show a clear bias in the model toward high-density, low-cost items. Table 3 shows the comparative repair costs ($icost_{t,y}$) of the different TAMCNs examined.

Table 3. TAMCNs Sorted by Repair Costs

TAMCN	Nomenclature	Repair Cost (Baseline)
A15037G	Air Search RADAR Set (AN/TPS-59A v3)	\$6,984,110.03
E18887M	Abrams Main Battle Tank	\$2,029,596.99
B01607B	Assault Breacher Vehicle	\$1,832,958.26
E08467K	Amphibious Assault Vehicle	\$1,034,127.03
B05897B	Armored Combat Earthmover	\$554,461.84
B00937B	Husky (Vehicle Mounted Mine Detection System)	\$533,152.01
E09427M	Light Armored Vehicle, Anti-Tank	\$509,020.64
D00527K	Armored Logistics Vehicle System Replacement	\$476,993.19
E06717M	Lightweight 155mm Howitzer	\$439,041.02
A21797G	Microwave Radio Terminal (AN/TRC-170)	\$362,768.02
D00037K	Armored Medium Tactical Vehicle Replacement	\$336,322.86
B00637B	Ruggedized Heavy Forklift (TRAM)	\$234,442.69
D00337K	Armored HMMWV	\$112,511.77
E14422M	M-16A4 Combat Rifle	\$1,086.96
E00177M	Rifle Combat Optic	\$371.63

This bias is partially explained by the objective formula:

$$\max \sum_{t,f,y} \text{value}_t \times \text{discount}^{y-y_0} \times \text{score}_{t,f,y} - \text{pen} \times \text{short}_{t,f,y} - 2 \times \text{pen} \times \text{cheat}_{t,f} - \text{pen} \times \text{aboveWmr}_{t,f,y}$$

This objective formula emphasizes the number of items repaired which is represented by a readiness score ($\text{score}_{t,f,y}$) as described in Chapter II. In the runs made for the initial DOE, the objective function value ranged between $-1,033,062$ and $-490,655$. The negative results demonstrate that not all items were repaired up to the prescribed readiness percentage, thus resulting in the imposition of a penalty term.

The penalty term itself must be examined. As described in Chapter II, the penalty term is formulated as:

$$\text{pen} = 2 \times \max_t \{\text{value}_t\}$$

This penalty term relies on the warfighting values assigned to each item, doubling the largest warfighting value. However, the warfighting values do not appear to provide enough of an incentive to focus on the repair of higher value items. Examination of how warfighting values are assigned is not included in the scope of this thesis. However, the use of warfighting values between 1 and 10 and using 15 significant digits to differentiate the different items clearly does not provide enough separation between the most and least valuable TAMCNs. This is clearly shown in Table 3, where the difference between the highest and lowest warfighting value is only 6.79 (with a ratio of 6.34). When treated as just a numeric, and not weighted significantly, this is not enough of a difference to force the model to emphasize higher value items. This is especially true when the difference in the baseline costs of the highest and lowest value items is over \$532,000 in favor of the lowest value item.

Table 4. TAMCNs Sorted by Warfighting Value

TAMCN	Nomenclature	Value
B00937B	Husky (Vehicle Mounted Mine Detection System)	8.06337431843702
E18887M	Abrams Main Battle Tank	7.14156939616450
B01607B	Assault Breacher Vehicle	6.84332166978724
B05897B	Armored Combat Earthmover	6.77287830598820
E09427M	Light Armored Vehicle, Anti-Tank	6.74086913194551
D00527K	Armored Logistics Vehicle System Replacement	6.49274816693941
A21797G	Microwave Radio Terminal (AN/TRC-170)	6.43325519808345
A15037G	Air Search RADAR Set (AN/TPS-59A v3)	6.42875058489059
E08467K	Amphibious Assault Vehicle	6.39707175577854
B00637B	Ruggedized Heavy Forklift (TRAM)	6.20388374152259
E06717M	Lightweight 155mm Howitzer	5.99437692376384
D00037K	Armored Medium Tactical Vehicle Replacement	5.68249367461217
D00337K	Armored HMMWV	4.92576372274664
E14422M	M-16A4 Combat Rifle	2.12111890843814
E0017A4	Rifle Combat Optic	1.26974935805521

The results of the stepwise regression support the hypothesis that the penalty term in the objective function is ineffective due to the small difference in warfighting values.

Despite the stepwise regression determining that the budget is a statistically insignificant estimator of the objective function, more analysis into budget changes on the model was needed. The objective function value was examined over the 11 budget changes in the DOE. An interesting result was seen; the objective function value decreased as the budget was increased as shown in Figure 3. This is an interesting result given that the budget is a constraint and it logically follows that when a constraint is loosened in a linear program the objective function value can only increase. Note: median counts were used during this analysis to decrease the likelihood of outlier interference in the analysis.

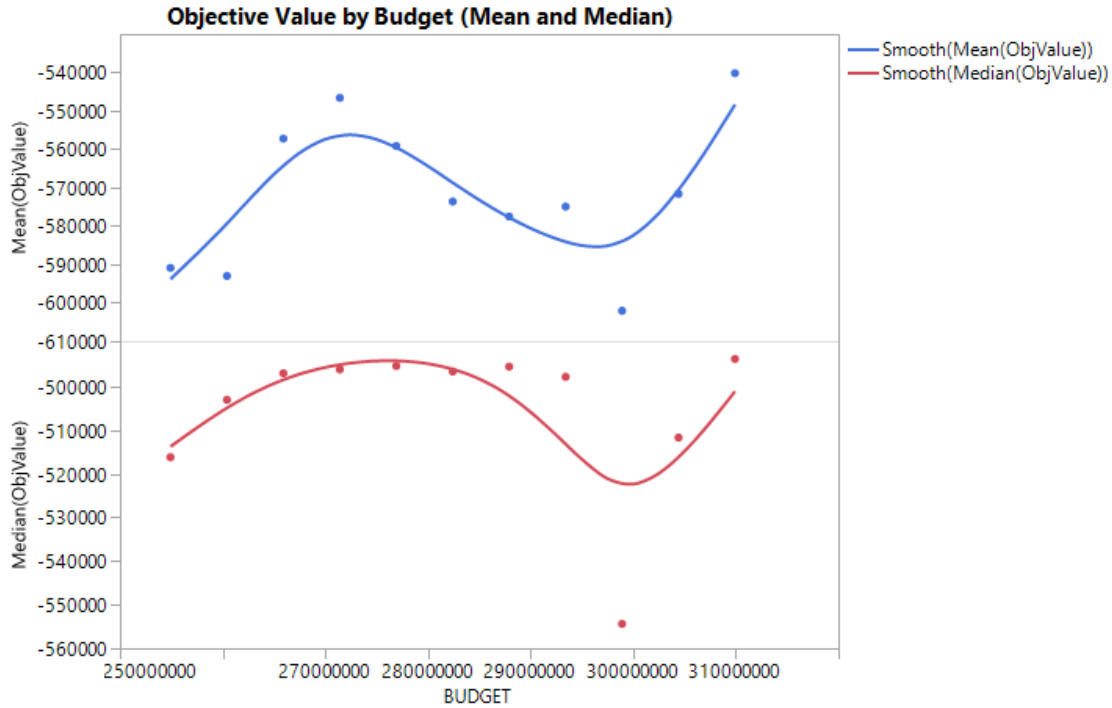


Figure 3. Mean and Median Objective Function Value by Budget

The behavior of the objective function value can be explained by the emphasis on warfighting value and the score variable. Once the budget rose enough the model was able to prioritize low-density, high-cost items that only required a few items repaired to maximize their score variable instead of relying on cheap, numerous repairs to maximize the objective function value. This is demonstrated in both Figures 4 and 5, where the five highest and lowest repair cost items are examined. It is shown that the number of low-cost items repaired was high and went higher as the budget was increased. However, it shows that the model emphasized the E18887M M1A1 Tank at first because of the high warfighting value. Once the budget increased slightly the emphasis went to the A15037G RADAR because it only required two to be repaired to maximize its score variable. This stayed consistent until the budget increased to over \$290 million at which point more M1A1s were repaired. This is also the budget value where the objective function decreased despite the loosening of the budget constraint.

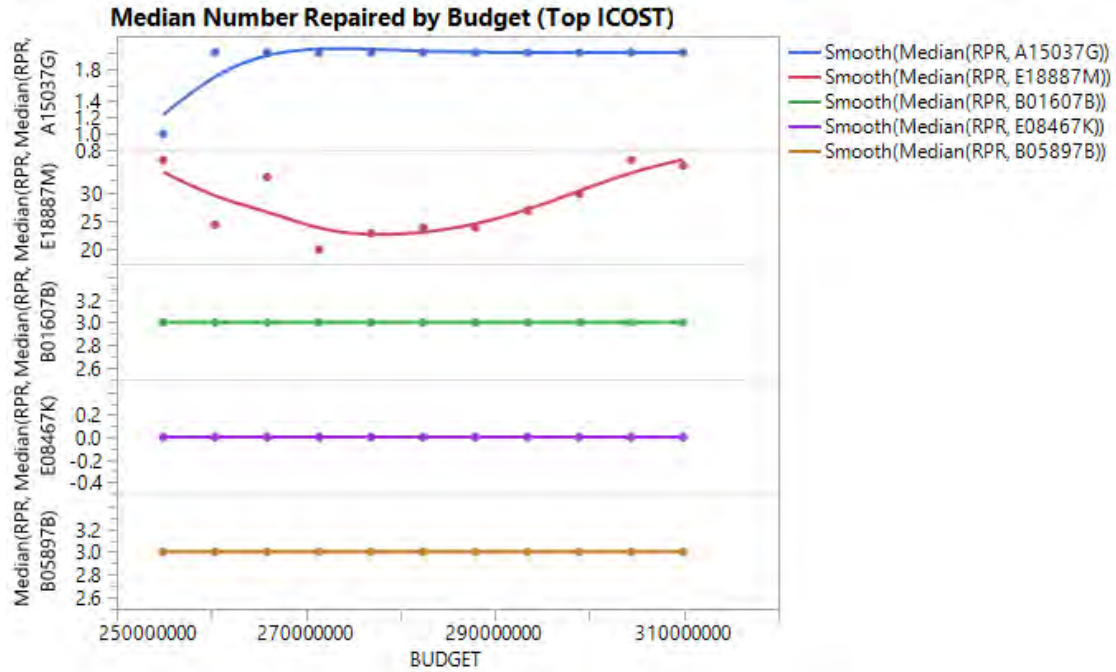


Figure 4. Median Number Repaired by Budget for the Five Highest ICOST Items

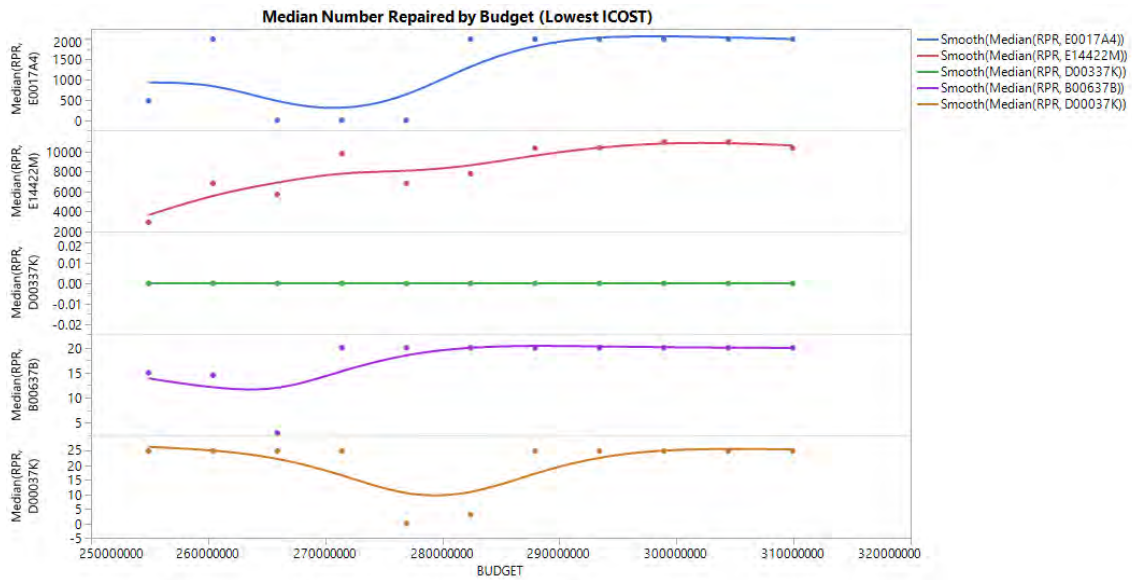


Figure 5. Median Number Repaired by Budget for the Five Lowest ICOST Items

The lowest cost items generally show what is to be expected. Due to their lower warfighting values the number of these items repaired increases as the budget is increased. This is only different for the D00037K MTRV, which shows a decrease around the \$280 million budget value.

In both Figures 4 and 5, there are items that show no change despite budget increases. In the case of the B0589 Armored Combat Excavator and B0160 Assault Breacher Vehicle, both have high warfighting values and only required three to be repaired. In every budget amount all the required items were repaired. The D0033 high-mobility, multi-purposed, wheeled vehicle, HMMWV, as shown in Figure 5, has a median value of zero repaired. This is explained by other variables. There is a requirement ($wmr_{t,f,y}$) of 1,429 serviceable D00337Ks, only 20 were turned in as unserviceable ($usr_{t,f,y}$), and the stratified ready to issue ($srfi_{t,f,y}$) was varied from 1404 to 1717. Even with the $rpct_{t,f,y}$ set at 95% the D00337K would only require two to be repaired. The mid-range warfighting value and high repair cost would make this a low priority item for the model. The E08467K Amphibious Assault Vehicle (AAV), shown in Figure 4, has a similar situation. With a requirement of 726 serviceable items and an SRFI varied from 684 to 834, the most that would ever need to be repaired would be six. Given the extremely high repair cost the model opted to fix a median of zero because the slight increase in the objective function value was not worth the expenditure.

B. WARFIGHTING RANKS

During analysis of the initial DOE, it appeared that the warfighting values were not different enough between the various items to cause sufficient prioritization in the model. To test this theory the warfighting values were replaced by ranks. This was done by ranking each TAMCN from 1 to 209, with 1 being the TAMCN with the lowest warfighting value and 209 the TAMCN with the highest warfighting value. Table 5 shows the warfighting ranks of the 15 TAMCNs used in the initial DOE.

Table 5. TAMCNs Sorted by Warfighting Rank

TAMCN	Nomenclature	Rank
B00937B	Husky (Vehicle Mounted Mine Detection System)	209
E18887M	Abrams Main Battle Tank	203
B01607B	Assault Breacher Vehicle	201
B05897B	Armored Combat Earthmover	198
E09427M	Light Armored Vehicle, Anti-Tank	195
D00527K	Armored Logistics Vehicle System Replacement	182
A21797G	Microwave Radio Terminal (AN/TRC-170)	178
A15037G	Air Search RADAR Set (AN/TPS-59A v3)	177
E08467K	Amphibious Assault Vehicle	175
B00637B	Ruggedized Heavy Forklift (TRAM)	166
E06717M	Lightweight 155mm Howitzer	157
D00037K	Armored Medium Tactical Vehicle Replacement	140
D00337K	Armored HMMWV	112
E14422M	M-16A4 Combat Rifle	4
E0017A4	Rifle Combat Optic	1

The intent of this change was to force the model to recognize significant differences between the highest and lowest value items. Further, it increases the penalty term from 16.2 to 418. Once these changes were made to the input file for ROMEfarmer the same DOE from before was used to generate another set of outputs.

The outputs from this experiment still show an overwhelming bias toward high-density, low-cost items. The same stepwise regression from the first experiment was run again with the new output. In this case, the stepwise results are extremely similar to the previous stepwise regression. However, there is evidence of some change, the 10th most influential predictor is now the interaction between the ICOST of the B0063 TRAM and SRFI of the E1888 M1A1, as shown in Figure 6.

Sorted Parameter Estimates					
Term	Estimate	Std Error	t Ratio		Prob> t
SRFI/E0017A4.ALL	106.21201	1.343e-6	7.9e+7		<.0001*
RPCT/E0017A4.ALL2020	-19979367	0.280511	-7e+7		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(SRFI/E0017A4.ALL-206801)	1824.8479	3.114e-5	5.9e+7		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(RPCT/E0017A4.ALL2020-0.84951)	-2.014e+8	4.091499	-5e+7		<.0001*
(RPCT/E14422M.ALL.2020-0.84971)*(SRFI/E14422M.ALL-115905)	1700.862	0.000038	4.5e+7		<.0001*
RPCT/E14422M.ALL2020	-8534703	0.214117	-4e+7		<.0001*
SRFI/E14422M.ALL	69.626302	2.351e-6	3e+7		<.0001*
(SRFI/E0017A4.ALL-206801)*(SRFI/E0017A4.ALL-206801)	-0.003716	1.38e-10	-3e+7		<.0001*
(RPCT/E14422M.ALL.2020-0.84971)*(RPCT/E14422M.ALL2020-0.84971)	-92660125	4.815914	-2e+7		<.0001*
(ICOST/B00637B.2020-234443)*(SRFI/E18887M.ALL-320)	-1.161624	6.75e-8	-2e+7		<.0001*

Figure 6. Top Ten Sorted Parameter Estimates for Stepwise Regression with Warfighting Ranks

While a more dramatic difference was expected, the stepwise regression model shows a step in the right direction by encouraging the repair of more high priority items. As before, the mean and median objective function values were examined over the range of budget changes. And, as before, a nearly identical trend is observed, as seen in Figure 7.

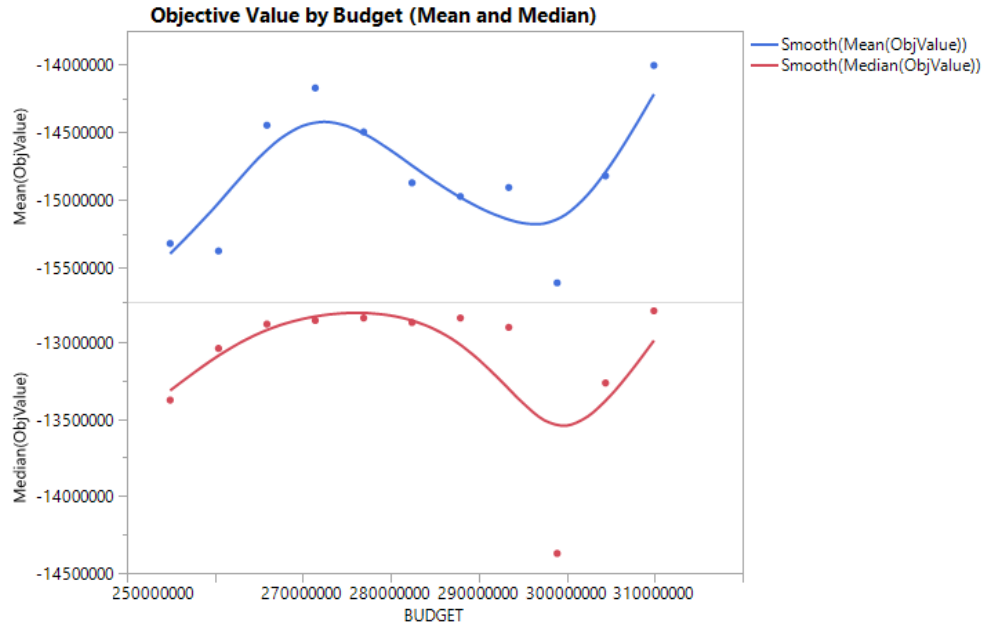


Figure 7. Objective Value by Budget with Warfighting Ranks

With little apparent change observed in the objective function value it was necessary to examine the individual TAMCN repairs conducted. A comparison was made between using the warfighting values and warfighting ranks for both the top and bottom five TAMCNs. The comparison between the top five TAMCNs shows little difference between using the warfighting values or ranks, as shown in Figures 8 and 9.

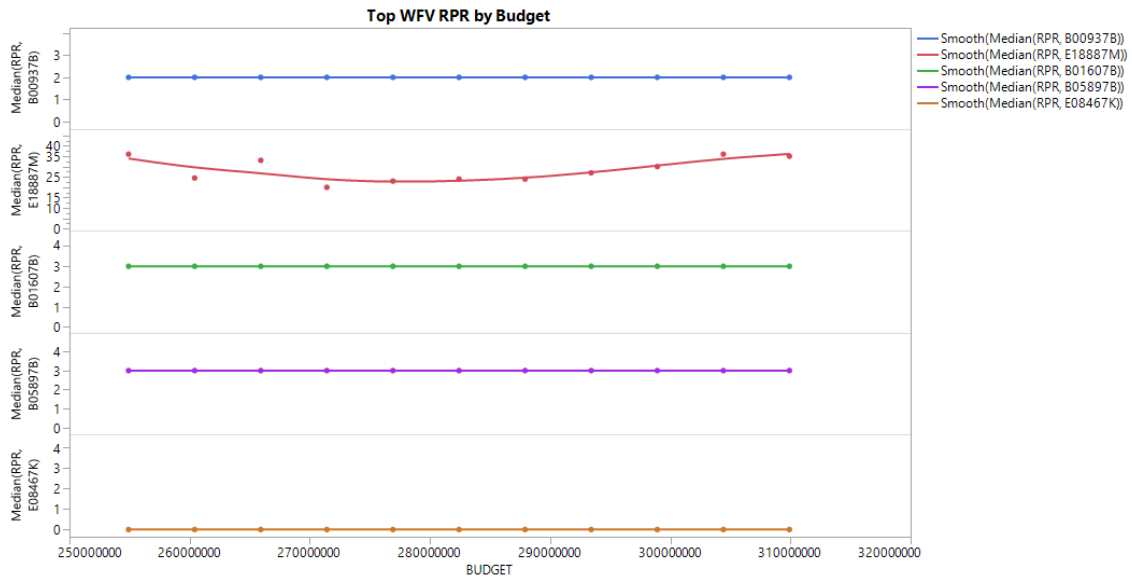


Figure 8. Median Number of Items Repaired of Each of the Top Five Warfighting Value TAMCNs by Budget

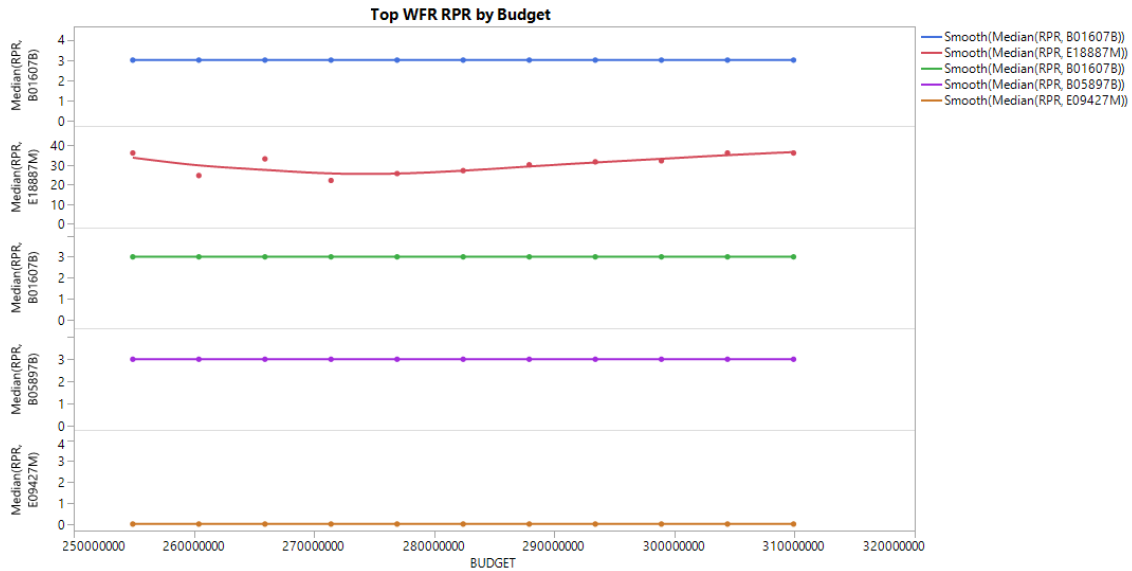


Figure 9. Median Number of Items Repaired of Each of the Top Five Warfighting Rank TAMCNs by Budget

The only change seen in the median number of items repaired from using warfighting values to ranks is that a single additional B0093 Husky is repaired when the warfighting ranks are used. While this does demonstrate more prioritized equipment being repaired, it does not appear to be the dramatic change that was expected. However, the dramatic change did not happen at the top of the prioritized list, but the bottom.

The use of warfighting ranks led to a significantly lower prioritization of the lower ranked TAMCNs. This is very apparent in Figures 10 and 11.

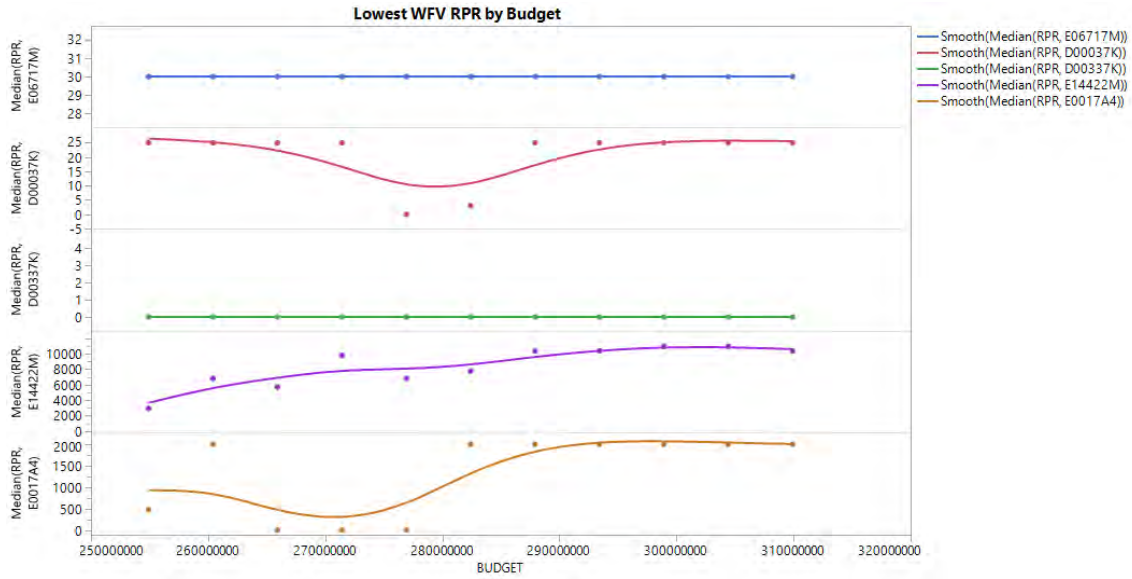


Figure 10. Median Number of Items Repaired of Each of the Bottom Five Warfighting Value TAMCNs by Budget

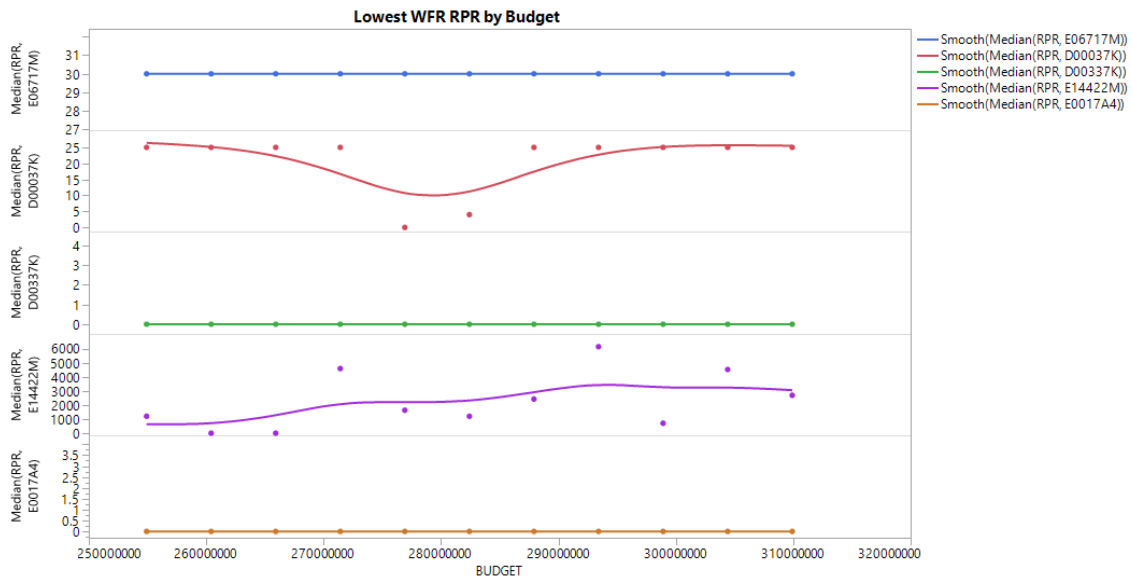


Figure 11. Median Number of Items Repaired of Each of the Bottom Five Warfighting Rank TAMCNs by Budget

When the original warfighting values were used, the median number of E0017 RCOs was zero for only three budget levels. However, when warfighting ranks are used,

the median number of RCOs repaired is zero regardless of the budget level. A similar result is seen with the E1442 M-16 rifle. When the warfighting values were used, the median number of rifles repaired was never below 2,900 with a maximum of 10,377. When the warfighting ranks were used, the median number repaired ranged from as low as zero for two budget levels to a maximum of 6,177. These dramatic changes demonstrate that by using the warfighting ranks instead of the warfighting values the model puts significantly more emphasis on repairing the higher priority items.

C. MAXIMIZING UNSERVICEABLE RETURNS BY TABLE OF AUTHORIZED MATERIEL CONTROL NUMBER

MARCORLOGCOM requested a final experiment that would examine what ROME would repair if every TAMCN had a large depot-level maintenance requirement at once. In order to simulate this, the unserviceable returns (USR) input file was altered to match the war materiel requirement (WMR) input file. This tells ROME that using units had turned in every item currently fielded for depot-level maintenance. Using the same DOE as in the first two experiments ROMEfarmer was run twice; once using the warfighting values and once using warfighting ranks. The outputs were analyzed as before. The results were surprising, insofar as they barely differed between the use of the warfighting values and ranks. The model's bias towards repairing low-cost, high-density items is far more noticeable during these ROMEfarmer runs than in earlier experiments.

Some noticeable changes are seen, specifically the results of the stepwise regression and the behavior of the objective function value as the budget was increased. In this experiment, the budget is the most critical parameter estimate in a stepwise regression that examined up to all two-way interactions and second order polynomials. This was true regardless if the warfighting values or ranks were used, as shown in Figures 12 and 13. Both regressions included over 200 terms and had an adjusted R-square of 1.00.

Sorted Parameter Estimates					
Term	Estimate	Std Error	t Ratio		Prob> t
BUDGET/ALL.2020	0.0146645	6.88e-10	2.1e+7		<.0001*
SRFI/E14422M.ALL	15.889465	1.287e-6	1.2e+7		<.0001*
RPCT/E0017A4.ALL.2020	-1792420	0.15023	-1e+7		<.0001*
RPCT/E14422M.ALL.2020	-1968744	0.220146	-9e+6		<.0001*
SRFI/E0017A4.ALL	8.1929148	9.195e-7	8.9e+6		<.0001*
ICOST/E0017A4.2020	-3241.444	0.000627	-5e+6		<.0001*
ICOST/E14422M.2020	-1224.873	0.000242	-5e+6		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(SRFI/E0017A4.ALL-206801)	46.574676	1.311e-5	3.6e+6		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(RPCT/E0017A4.ALL.2020-0.84951)	-5146242	2.927732	-2e+6		<.0001*
(SRFI/E0017A4.ALL-206801)*(SRFI/E0017A4.ALL-206801)	-0.000105	1.03e-10	-1e+6		<.0001*

Figure 12. Top Ten Sorted Parameter Estimates of the Stepwise Regression Results using Warfighting Values from JMP

Sorted Parameter Estimates					
Term	Estimate	Std Error	t Ratio		Prob> t
BUDGET/ALL.2020	0.3802439	5.098e-9	7.5e+7		<.0001*
RPCT/E14422M.ALL.2020	-51024501	1.052405	-5e+7		<.0001*
RPCT/E0017A4.ALL.2020	-46443554	0.99786	-5e+7		<.0001*
SRFI/E14422M.ALL	411.95508	1.119e-5	3.7e+7		<.0001*
ICOST/E14422M.2020	-31794.57	0.000911	-3e+7		<.0001*
SRFI/E0017A4.ALL	212.20577	6.72e-6	3.2e+7		<.0001*
ICOST/E0017A4.2020	-83981.96	0.003137	-3e+7		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(SRFI/E0017A4.ALL-206801)	1209.5415	9.462e-5	1.3e+7		<.0001*
(RPCT/E0017A4.ALL.2020-0.84951)*(RPCT/E0017A4.ALL.2020-0.84951)	-1.324e+8	19.43212	-7e+6		<.0001*
(SRFI/E0017A4.ALL-206801)*(SRFI/E0017A4.ALL-206801)	-0.002795	5.65e-10	-5e+6		<.0001*

Figure 13. Top Ten Sorted Parameter Estimates of the Stepwise Regression Results using Warfighting Ranks from JMP

However, like all previous stepwise regressions, it is clear that the high-density, low-cost items have an outsized influence on the model compared to other TAMCNs. This is clearly demonstrated by how all but one of the top ten parameter estimates are related to the M-16 and RCO.

Unlike previous experiments, the objective function value rose in a nearly linear fashion as budget increased, as shown in Figures 14 and 15.

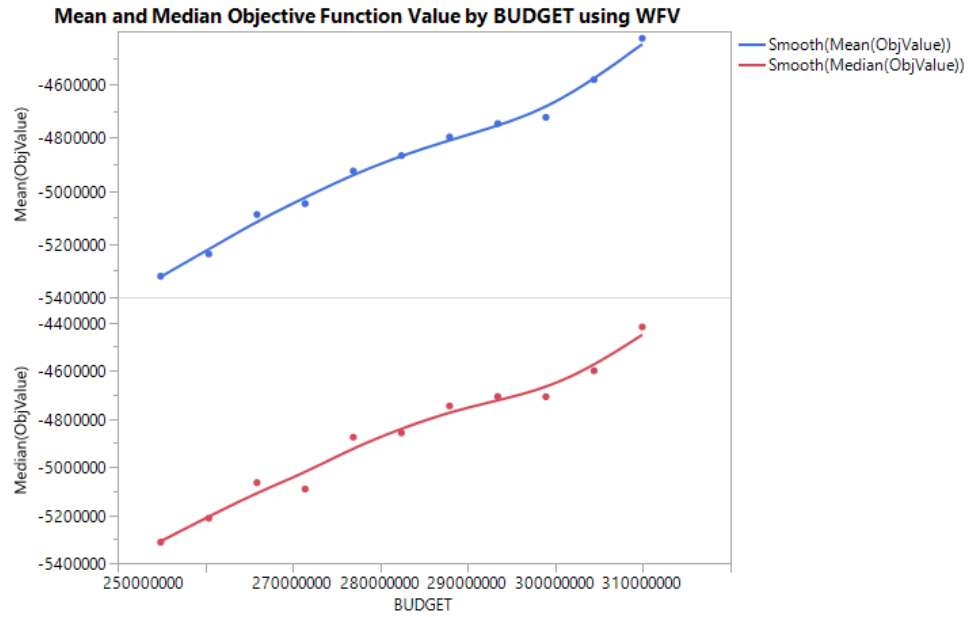


Figure 14. Mean and Median Objective Function Value by Budget using Warfighting Value

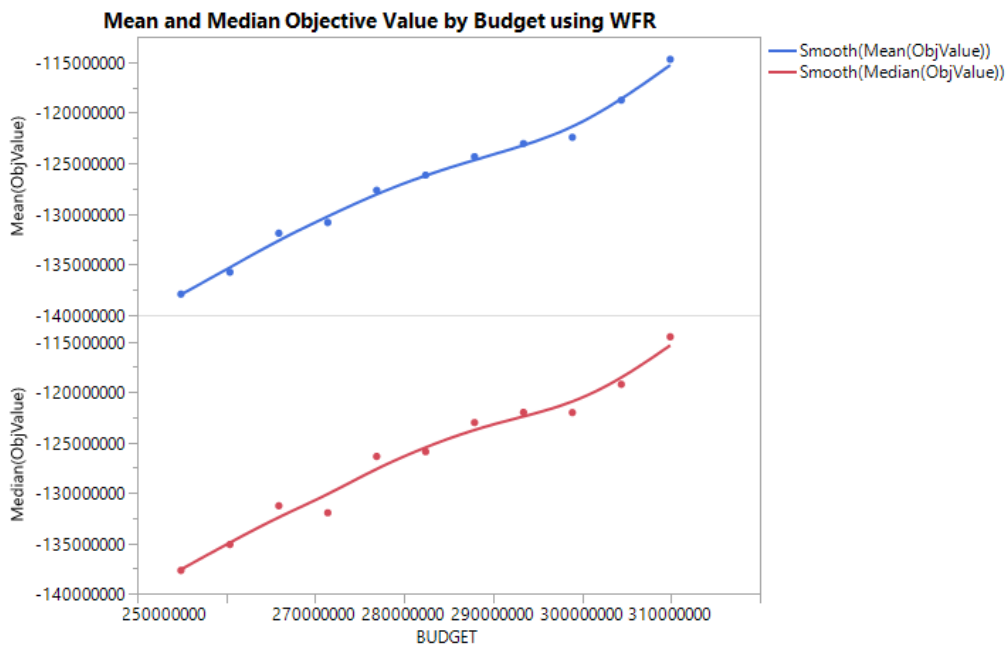


Figure 15. Mean and Median Objective Function Value by Budget using Warfighting Rank

This implies that the mixed integer linear program is behaving in an expected manner by increasing the objective function value as a constraint is loosened. This behavior is different from what was observed in previous experiments because of the vast number of possible repairs the model can select from. When USRs for many of the low-density TAMCNs were much lower, there were fewer options to be repaired to increase the objective function value.

The most dramatic results from these two ROMEfarmer runs are seen in the median number of each TAMCN repaired when all TAMCNs require significant investment to meet readiness goals. The model's bias toward high-density, low-cost items is starkly demonstrated in these experiments. In both instances, using the warfighting values and warfighting ranks, the top priority TAMCNs have a median repair number of zero as shown in Figures 16 and 17.

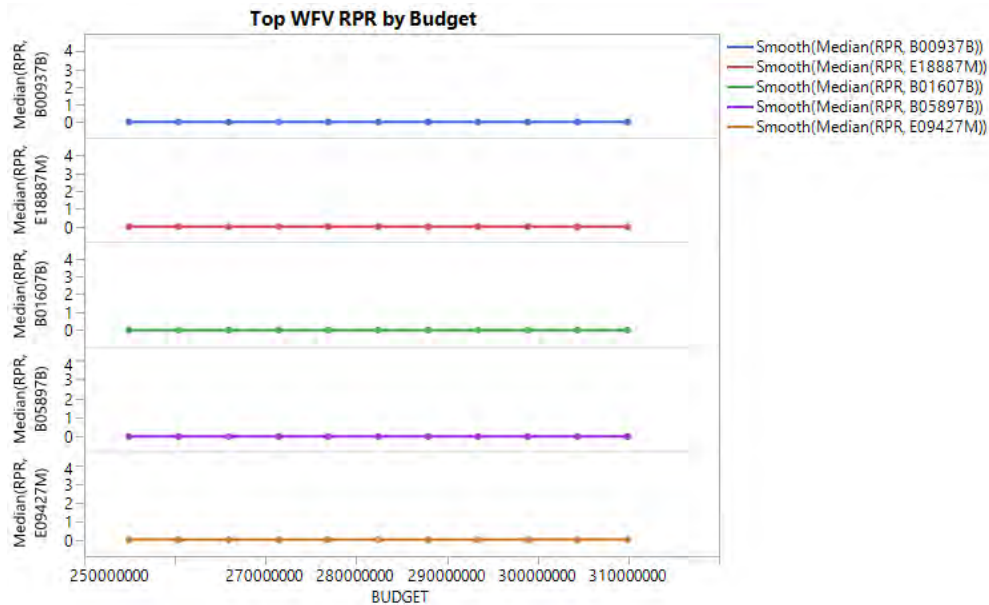


Figure 16. Median Items Repaired for Top Five Warfighting Value TAMCNs

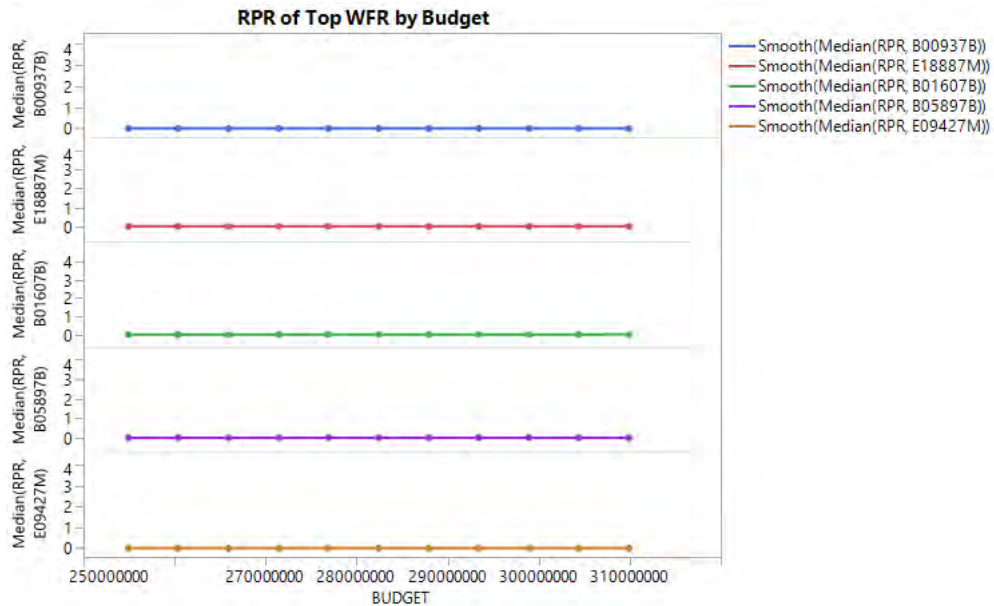


Figure 17. Median Items Repaired for Top Five Warfighting Ranks TAMCNs

While none of the highest valued TAMCNs were repaired a great many of the two lowest valued TAMCNs were. As shown in Figures 18 and 19, the M-16 and RCO were repaired at incredibly high rates compared to all other TAMCNs.

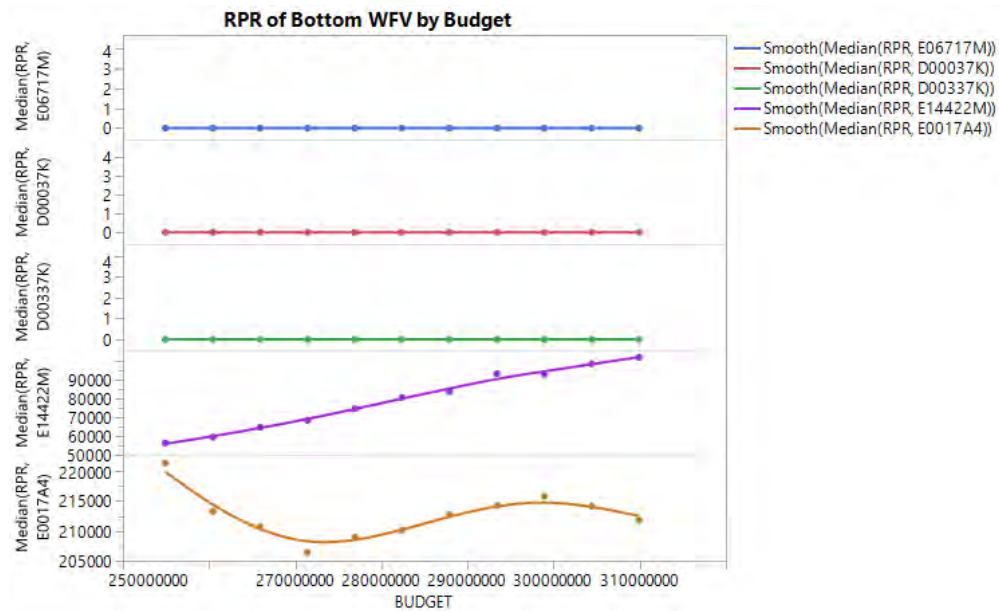


Figure 18. Median Items Repaired for Bottom Five Warfighting Values TAMCNs

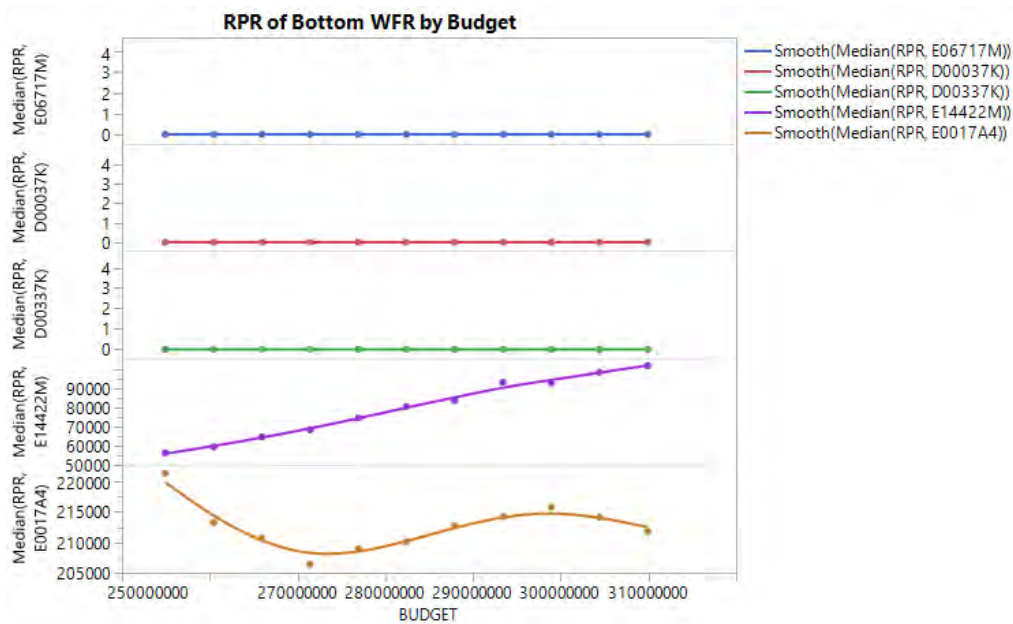


Figure 19. Median Items Repaired for Bottom Five Warfighting Ranks TAMCNs

What these results state, simply, is that if nearly everything in the Marine Corps broke at the same time, the ROME model would only repair M-16s and RCOs in practically significant numbers. A primary reason for this, as already described in this chapter, is the lack of significant differences between the warfighting values of the various TAMCNs. Warfighting ranks were introduced to attempt to remedy this issue; however, these final two runs of ROMEfarmer demonstrate that even the warfighting ranks do not provide enough of a difference to sufficiently impact the objective function. This is very apparent when both the ranks and values are examined as a ratio of repair cost per warfighting unit. This was calculated in the following way:

$$\frac{\text{value}_t}{\text{icost}_{t,y}} = \text{warfighting value per dollar of repair cost}$$

$$\frac{\text{rank}_t}{\text{icost}_{t,y}} = \text{warfighting rank per dollar of repair cost}$$

As Table 5 shows, the differences between the various TAMCNs is incredibly close, with the exceptions of the M-16 and RCO, which enjoy a nearly hundredfold advantage. Thus, they dominate what ROME recommends to repair.

Table 6. TAMCNs Sorted by Warfighting Value per Dollar of Repair Cost

TAMCN	Nomenclature	WFV	WFR	ICOST	WFV per Cost	WFR per Cost
E0017	RCO	1.26974935805521	1	371.63	0.003416724	0.002690865
E1442	M-16A4	2.12111890843814	4	1086.96	0.001951415	0.003679974
D0033	M1152 HMMWV	4.92576372274664	112	112511.77	0.000043780	0.000995451
B0063	TRAM	6.20388374152259	166	234442.69	0.000026462	0.000708062
A2179	AN/TRC-170	6.43325519808345	178	362768.02	0.000017734	0.000490672
D0003	MTVR	5.68249367461217	140	336322.86	0.000016896	0.000416267
B0093	Husky	8.06337431843702	209	533152.01	0.000015124	0.000392008
E0671	Howitzer, M777	5.99437692376384	157	439041.02	0.000013653	0.000357598
D0052	LVSR	6.49274816693941	182	476993.19	0.000013612	0.000381557
E0947	LAV	6.74086913194551	195	509020.64	0.000013243	0.000383089
B0589	M9 ACE	6.77287830598820	198	554461.84	0.000012215	0.000357103
E0846	AAV	6.39707175577854	175	1034127.03	0.000006186	0.000169225
B0160	Assault Breacher Vic	6.84332166978724	201	1832958.26	0.000003733	0.000109659
E1888	M1A1 Tank	7.14156939616450	203	2029596.99	0.000003519	0.000100020
A1503	Air Search RADAR	6.42875058489059	177	6984110.03	0.000000920	0.000025343

The lack of appreciable differentiation between the various TAMCNs demonstrates the issue with using twice the highest warfighting value as a penalty term. It is much too easily overcome by numbers of RCOs and M-16s that are needed to be repaired. When rank is used the differences are negligible given how small these ratios are. This analysis also suggests the process for developing warfighting values be reassessed, as it does not seem intuitive that an M1A1 tank would have only six times the warfighting value of a rifle scope.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND RECOMMENDATIONS

A. STUDY CONCLUSIONS

The primary conclusion of this thesis is that the current objective function value in ROME inadequately prioritizes the allocation of repair dollars. Ostensibly, the use of the warfighting values is meant to ensure higher priority items are allocated repair dollars before lower priority items. However, as demonstrated through extensive testing using an advanced design of experiments, the model did not react as expected. The current system employed by MARCORLOGCOM to overcome this issue is the “fencing off” of priority equipment. This is usually done by SMEs who choose which systems will be given full funding before the model is even run. This is done because these same SMEs understand that the model does not provide the desired outcomes when left to run on its own. This thesis has found that while the current ROME model does save hundreds of man-hours that would be spent manually calculating the appropriation of hundreds of millions of dollars across 209 TAMCNs it does not do so to the satisfaction of MARCORLOGCOM. Further, the ROME model has such a bias toward high-density, low-cost items that it cannot be overcome even by drastically changing the warfighting values (e.g., to warfighting ranks).

B. RECOMMENDATIONS

The following recommendations are broken into two parts, the first being recommendations to MARCORLOGCOM to improve the ROME model and the second being recommendations to the larger Marine Corps to improve TAMCN prioritization. These recommendations are based on the extensive testing described throughout this thesis and the author’s fleet experience and formal study of Marine Corps doctrine.

1. Marine Corps Logistics Command Recommendations

Additional study of the ROME model by MARCORLOGCOM is required. The focus of this additional study should center on three critical points. The first two concern the formulation of the ROME model itself and the third is coding of the model.

Throughout this thesis, it has been demonstrated that the objective function has a bias towards allocating funds to high-density, low-cost TAMCNs. Analysis of the experiments conducted point to a too weak penalty term and insufficient differentiation in warfighting values as the likely causes of this bias. It is recommended that the objective function be reformulated with a much harsher penalty term for not repairing higher priority items first. Once a reformulation is done, it should be tested using ROMEfarmer to ensure that the proper changes have been made and the results are more intuitive. This will likely be an iterative process by which changes can be made in a controlled manner and then tested with an advanced DOE to ensure the results are what is desired by MARCORLOGCOM. While this recommendation will likely improve the results of the ROME model, it will likely never eliminate the need for SMEs to examine the results and make changes based on repair facility capabilities and ever-changing Marine Corps priorities.

Further, an item not examined in this thesis, is the incorporation of a time constraint. Currently, the ROME model does not take into account how much time is needed to repair an individual TAMCN item. This has an obvious impact as to how funds can be allocated. Take, for example, the massive tornado that struck the depot maintenance facility in Albany, GA in January of 2017. Hundreds, if not thousands, of man-hours were lost due to massive damage to maintenance bays. If the ROME model had been re-run after this incident to determine where funds should be reallocated no consideration would have been given to the decreased time available to conduct repairs. It is recommended that a time-to-task study be conducted at MARCORLOGCOM to add a time variable so the model will not allocate funding to TAMCN items that do not have the time to be repaired in the fiscal year the model is allocating for.

Finally, MARCORLOGCOM should invest in recoding the model into a free optimization coding language, such as Pyomo. Currently, the ROME model is coded in the General Algebraic Modeling System, GAMS, an optimization software that costs the Marine Corps at least \$3,200 a year to procure a license for a single user (GAMS 2019). While this was a wise choice when the model was created, as free optimization coding languages had not advanced to the point of equity with proprietary options, this is no longer the case. Pyomo, specifically, has advanced to the point that optimization experts

at NPS now teach students to optimize using it instead of expensive proprietary options such as GAMS.

2. Marine Corps Recommendations

The following recommendations concern depot-level maintenance, but require decisions to be made outside of MARCORLOGCOM. These two recommendations concern the prioritization of TAMCN items for depot-level maintenance allocations. The first is the use of the DRACM model to establish warfighting values and the second is how the DRACM results are certified.

The current DRACM model output is the warfighting values that are fed into the ROME model. The current warfighting values cover 209 TAMCNs, but with only a ratio difference between the highest and lowest priority TAMCNs of 6.34. This is not nearly enough value differential between TAMCNs for ROME to make meaningful differentiation between the high and low priority TAMCNs. Without any changes to the objective function a change needs to be made to the DRACM model to ensure that high-priority TAMCNs are given the prioritization they need. This thesis, conclusively demonstrates that rankings in lieu of values, while a good first step, is not enough. A study into how to increase the effectiveness of the DRACM model should be undertaken to improve the depot-level maintenance budget allocation process. Given the current system of the warfighting values being certified by the Deputy Commandant for Plans, Policies, and Operations (PP&O) any changes to the DRACM model will need to be verified by both MARCORLOGCOM and PP&O.

Finally, the Marine Corps, as a whole, needs to examine how TAMCNs are prioritized by DRACM. Currently, the highest priority TAMCN that is handled by MARCORLOGCOM is the B0093 Husky, a vehicle that was designed for the sole-purpose of finding improvised explosive devices (IEDs) in a desert environment. This prioritization does not seem to fit the new priorities set out by the Commandant in the 2016 Marine Operating Concept (Headquarters United States Marine Corps 2016). If the Marine Corps is expected to begin to confront near-peer and peer rivals, it must emphasize the equipment that will be needed in those types of fights.

APPENDIX A. ROME USER'S MANUAL

Repair Optimization and Materiel Evaluator (ROME) User's Manual

Major Antonio DeMarco
Operations Research Analyst
MARCORLOGCOM LCC
Studies and Analysis Division

06 Oct 2014

Contents

1	Introduction	2
1.1	Problem Statement	2
1.2	Intended Use and Application	2
1.3	Model History and Overview	3
2	Mathematical Formulation of the Optimization Program	5
2.1	Index Sets	5
2.2	Parameters	5
2.3	Variables	6
2.4	Objective Function	7
2.5	Constraints	8
3	Architecture	11
3.1	Prerequisites	11
3.2	Directory Structure	11
4	Preparing to Run the Model	13
4.1	Inputs	14
4.2	Some Notes on Inputs	16
5	Running the Model	21
5.1	Output	25
6	Capabilities, Limitations and Assumptions	26
6.1	Capabilities	26
6.2	Limitations	26
6.3	Assumptions	26
7	Configuration Management	29

1 Introduction

This User's Manual is intended to assist analysts employed in the Studies and Analysis Division, Operations Research Branch of Marine Corps Logistics Command to understand, operate and update the ROME model.

1.1 Problem Statement

The United States Marine Corps (USMC) uses equipment to execute its warfighting mission. In May 2011, the Marine Corps possessed approximately seven million Table of Materiel Control Number (TAMCN) ground equipment items. Much of this equipment is consumable and is disposed of when it reaches its life limit. Other items are repairable and undergo maintenance actions to return the items to usable status. These repairable items are often larger and more expensive than consumables.

Funds are limited for maintenance and the Marine Corps is often faced with the challenge of judiciously choosing to repair only the most mission critical equipment. As a result, some nonessential equipment repairs are delayed. These choices are made more difficult by the many statutory and service mandated requirements. However, optimization algorithms have the computational strength to consider these many competing demands to identify the most beneficial equipment repair scheduling.

1.2 Intended Use and Application

One tool in the Enterprise Lifecycle Maintenance Program (ELMP) process is the Repair Optimization Materiel Evaluator (ROME). The ROME evaluates repairable equipment, considering: budget constraints, cost of repair, importance of the equipment to the Marine Corps (evaluated on warfighting values derived from the Dimension Reduction And Clustering Means (DRACM) model), future asset procurement plans, and target readiness (i.e. decreased quantity of on-hand deadlined equipment); to provide an optimal mix of equipment for repair. The output provided by the ROME is then evaluated by lifecycle asset manager subject matter experts (SME) and a final plan is developed. This plan aids decision makers in the development of defensible Program Objective Memorandum (POM) submissions by balancing competing statutory mandates, service mandates, and fiscal constraints.

The Repair Optimization Materiel Evaluator (ROME) portion of the ELMP is designed to:

- Help lifecycle asset managers satisfy the many competing objectives (e.g., affordability, operational availability, maintenance line availability, and the need to increase Marine Corps readiness ratings) to find the proper mix of equipment to induct into depot maintenance.

- Adapt to changing circumstances. The tool allows users to alter model input parameters and constraints to appropriately reflect changing circumstances (e.g., budget increases or decreases, changing equipment priorities, changes in availability of equipment).
- Enable planners and analysts to compare multiple scenarios and assess the impacts of different assumption sets (i.e., different model input parameters and constraints) against multiple competing priorities (e.g., affordability, operational availability, 50/50 and Core).
- Enable rapid setup, execution of model runs, and analysis of model output in support of rapid what-if analyses.
- Assist lifecycle asset managers with producing repeatable, auditable, quantitative input to Program Objective Memorandum (POM) and budget submissions.

Output from the ROME produces the following:

- An optimal list of equipment repair.
- Reports supporting POM submissions.
- Reports addressing initial on-hand ready for issue (RFI) equipment, on-hand not ready for issue (NRFI) equipment, TAMCN list of interest, warfighting values, procurement of each TAMCN during the evaluation period, target readiness values for each TAMCN, and the cost of repair for each piece of equipment.

The ROME is used to justify funding requirements submitted through the POM and the policy, processes, and transactional automated information systems used to execute those funds provided in response to POM submissions. MARCORLOGCOM continuously assesses the ROME against the VV&A requirements and criteria. The results of this continued evaluation, as well as any modifications to the ROME, related policy, processes or authoritative data sources will be included in future updates to this document.

1.3 Model History and Overview

Since 1999, Marine Corps Logistics Command (MARCORLOGCOM) has been using the Depot Level Maintenance Program (DLMP) to assist life cycle asset managers in determining the appropriate mix, quantities, and funding levels needed to maintain USMC ground equipment. Due to long running contingency operations, changes in policy, and modernizations in automated information systems, stakeholders became increasingly dissatisfied with the DLMP. MARCORLOGCOM leadership considered customer feedback and decided to reengineer the DLMP to comply with policy and better satisfy the needs of the Marine Corps. The new process is known as ELMP. One of the consequences of these changes was the requirement to review and reformulate the optimization model used by the DLMP, known as the Dynamic Equipment Repair Optimization (DERO).

The new optimization model, known in the ELMP order as the Enterprise Lifecycle Maintenance Program Price and Performance Model (ELMP PPM), implements all required changes and builds on the DERO model. The updated model eliminates unnecessary sub-routines and strengthens essential preexisting logic. Automation in data mining, data input, and resulting exhibits has been enhanced and are closely aligned to DOD policies.

ROME is the name given to the ELMP PPM by its creators. At its heart is a mixed integer programming (MIP) model. In 1999, Captain C.A. Goodhart, USMC at DC/S Installations and Logistics (LX) developed the DERO tool. The current ROME model is based on the work of Goodhart. With careful attention to formulation of constraints, Major Antonio DeMarco USMC working at the Studies and Analysis (S&A) Division, MARCORLOGCOM, has built upon the foundation of DERO and the work of CSA/Palladium on the code of DERO, to develop the ROME, the newest version of the optimization tool.

The optimization routine in ROME is coded using the General Algebraic Modeling System (GAMS), a commercial-off-the-shelf (COTS) high-level modeling system. MARCORLOGCOM has also purchased three linear and nonlinear solvers to complement GAMS: CPLEX, CONOPT and MINOS.

2 Mathematical Formulation of the Optimization Program

This section contains a description of the mixed integer program in NPS standard format.

2.1 Index Sets

- $s \in S$ (e.g. $S = \{1, 2, 3, 4, 5\}$): indices for the lines which bound the $\text{score}_{t,f,y}$ variable used in the objective function (read in from the `SEGMTSET.txt` file)
- $t \in T$ (e.g. $T = \{A0004, A0017\}$): TAMCNs of interest (read in from the `TAMCNSET.txt` file)
- $f \in F$ (e.g. $F = \{\text{ACTIVE}, \text{RESERVES}\}$): units of interest (read in from the `FORCESET.txt` file)
- $y \in Y$ (e.g. $Y = \{2010, 2011, \dots, 2017\}$): years of interest (read in from the `YEARSET.txt` file)

2.2 Parameters

- $\text{wmr}_{t,f,y}$: war materiel requirement for t from f during y (read in from the `WMR.txt` file)
- $\text{usr}_{t,f,y}$: unserviceable returns that are forecast to occur of t from f during y (read in from `USR.txt` file)
- $\text{wo}_{t,f,y}$: washouts that are forecast to occur of t from f during y (read in from `WO.txt` file)
- $\text{rtgt}_{t,f,y}$: readiness target of t for f during y (read as a fraction from the `RPCT.txt` file and later transformed into an integer value representing the number of units)
- $\text{icost}_{t,y}$: repair cost per unit for t during y (read in from the `ICOST.txt` file)
- $\text{issue}_{t,y}$: number of t forecast to be issued during y from new procurement (read in from the `ISSUE.txt` file)
- $\text{budget}_{f,y}$: the budget for f during y (read in from the `BUDGET.txt` file)
- $\text{srfi}_{t,f}$: starting number of stratified rfi t at f in year 0 (read in from the `SRFI.txt` file)
- $\text{sbl}_{t,f}$: starting number of stratified nrfi t at f in year 0 (read in from the `SBL.txt` file)

- $uspare0_t$: number of rfi t not stratified (excess) in year 0 (read in from the `USPARE0.txt` file)
- $dspare0_t$: number of nrfi t not stratified (excess) in year 0 (read in from the `DSPARE0.txt` file)
- $unfF0_{t,f}$: number of condition code F t at f in year 0 (read in from the `UNFF0.txt` file)
- $value_t$: warfighting value of t , allows for subjectively placing more or less value on each TAMCN (read in from `value.txt` file)
- $slope_s$: slope of line s used in bounding the $score_{t,f,y}$ variable (read in from the `SLOPE.txt` file)
- $intercept_s$: intercept of line s used in bounding the $score_{t,f,y}$ variable (read in from the `INTCPT.txt` file)
- $discount$: fractional value of repairing an item in year y versus year $y - 1$ (read in from `discount.txt` file)
- $limRpr$: binary set to 1 if rpr is limited from above to usr
- $fixFund$: binary set to 1 if a funding minimum is to be set
- $minFund$: the fractional amount of the max funding to use for the minimum
- y_0 : the first year (taken as the earliest year in Y)
- $tilim_{t,f,y}$: the max allowed number of t which may become excess from f during y (calculated in the `calcddata.txt` script)

This parameter is used in the logic which regulates the transfer of items to and from excess. It is calculated as follows.

$$\begin{aligned} tilim_{t,f,y} &= \max\{0, wmr_{t,f,y-1} - wmr_{t,f,y}\} \quad \forall t, f, y | y > y_0 \\ tilim_{t,f,y_0} &= \max\{0, srfi_{t,f} + sbl_{t,f} - wmr_{t,f,y_0}\} \quad \forall t, f \end{aligned}$$

- pen : a penalty for being short of $rtgt_{t,f,y}$, for using $cheat_{t,f}$ variable or for being in excess of $wmr_{t,f,y}$

$$pen = 2 \cdot \max_t \{value_t\}$$

2.3 Variables

- $score_{t,f,y}$: a measure of the readiness of t in f during y
- $rpr_{t,f,y}$: the number of t funded for repair by f during y
- $rfi_{t,f,y}$: the number of rfi t in f at end of y

- $\text{nrfi}_{t,f,y}$: the number of rfi t in f at end of y
- $\text{isrfi}_{t,y}$: excess rfi t at end of y
- $\text{isnrfi}_{t,y}$: excess nrfi t at end of y
- $\text{unfF}_{t,f,y}$: c/c F t in f at end of y
- $\text{tis}_{t,f,y}$: number of rfi t which became excess in f during y
- $\text{tiu}_{t,f,y}$: number of nrfi t which became excess in f during y
- $\text{float}_{t,f,y}$: excess rfi t redistributed to f during y
- $\text{strn}_{t,f,y}$: excess nrfi t redistributed to f during y
- $\text{tedef}_{t,f,y}$: the total number of t below wmr in f at the end of y
- $\text{defind}_{t,f,y}$: a binary variable set to 1 if $\text{tedef}_{t,f,y} > 0$
- $\text{cheat}_{t,f}$: variable to allow feasibility in case of poor forecasting, these represent t which had to be stratified to f in the beginning to meet the unserviceable returns forecast
- $\text{short}_{t,f,y}$: the number of t below the readiness target for f at the end of y
- $\text{excess}_{t,f,y}$: the number of t above the readiness target for f at the end of y
- $\text{aboveWmr}_{t,f,y}$: the number of t above the wmr target for f at the end of y

2.4 Objective Function

WTDVALUE

$$\begin{aligned} \text{maximize } \sum_{t,f,y} & (\text{value}_t \text{discount}^{y-y_0} \text{score}_{t,f,y} - \text{pen} \cdot \text{short}_{t,f,y} \\ & - 2 \cdot \text{pen} \cdot \text{cheat}_{t,f} - \text{pen} \cdot \text{aboveWmr}_{t,f,y}) \end{aligned}$$

- basic idea is to maximize readiness (represented by $\text{value}_t \text{score}_{t,f,y}$)
- multiplying the readiness term by discount^{y-y_0} encourages earlier repair of assets
- the negative terms penalize being short of the readiness goal, having to “cheat” by adding rfi items up front to ensure feasibility and holding excess above wmr

2.5 Constraints

BUDLIM

$$\sum_t \text{icost}_{t,y} \text{rpr}_{t,f,y} \leq \text{budget}_{f,y} \quad \forall f, y$$

- simple budget constraint

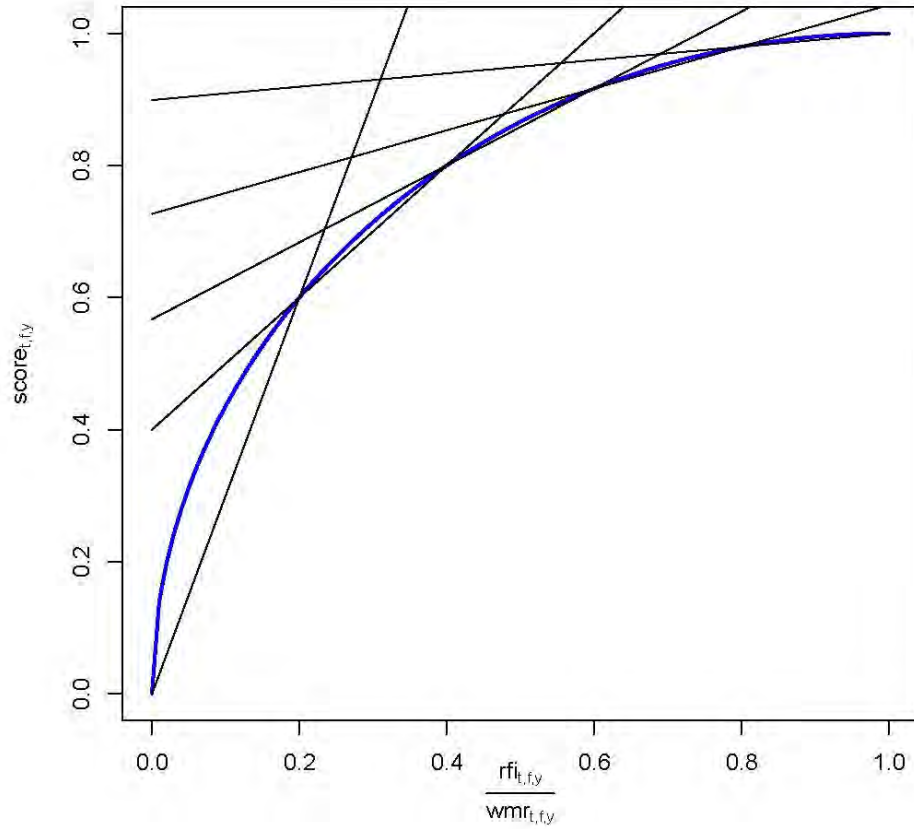
FUNDMIN

$$\sum_t \text{icost}_{t,y} \text{rpr}_{t,f,y} \geq \text{budget}_{f,y} \cdot \text{minFund} \cdot \text{fixFund} \quad \forall f, y$$

- sets a minimum funding amount

CALCSCORE

$$\text{score}_{t,f,y} \leq \begin{cases} \text{intcpt}_s + \text{slope}_s \frac{\text{rfi}_{t,f,y}}{\text{wmr}_{t,f,y}} & \text{wmr}_{t,f,y} > 0 \\ 0 & \text{wmr}_{t,f,y} = 0 \end{cases} \quad \forall s, t, f, y$$



The lines (which approximate the curve shown in blue) bound the $\text{score}_{t,f,y}$ variable from above. The effect is that a marginal increase in readiness will lead to a greater marginal increase in $\text{score}_{t,f,y}$ when readiness is lower.

RDYGOAL

$$\text{rfi}_{t,f,y} = \text{rtgt}_{t,f,y} - \text{short}_{t,f,y} + \text{excess}_{t,f,y} \quad \forall t, f, y$$

- establishes the values for the $\text{short}_{t,f,y}$ and $\text{excess}_{t,f,y}$ variables

WMRBAL

$$\text{rfi}_{t,f,y} + \text{nrfi}_{t,f,y} = \text{wmr}_{t,f,y} - \text{tedef}_{t,f,y} + \text{aboveWmr}_{t,f,y} \quad \forall t, f, y$$

- establishes the value for the $\text{tedef}_{t,f,y}$ and $\text{aboveWmr}_{t,f,y}$ variables, these are the amount of t short of and above the war materiel requirement for f in y respectively

PREVTI1A

$$\text{tedef}_{t,f,y} \leq \text{defind}_{t,f,y} \text{wmr}_{t,f,y} \quad \forall t, f, y$$

- along with PREVTI1B, establishes the value for the $\text{defind}_{t,f,y}$ variable, this is the binary variable which is set to 1 if the number of t is short of wmr in f for y

PREVTI1B

$$\text{tedef}_{t,f,y} \geq \text{defind}_{t,f,y} \quad \forall t, f, y$$

- along with PREVTI1A, establishes the value for the $\text{defind}_{t,f,y}$ variable, this is the binary variable which is set to 1 if the number of t is short of wmr in f for y

PREVTI2

$$\text{tiu}_{t,f,y} + \text{tis}_{t,f,y} \leq \text{tilim}_{t,f,y}(1 - \text{defind}_{t,f,y}) \quad \forall t, f, y$$

- establishes an upper bound on the number of t that can become excess from f during y

UPPOOL

$$\text{isrfi}_{t,y} = \text{isrfi}_{t,y-1} + \text{issue}_{t,y} + \sum_f (\text{tis}_{t,f,y} - \text{float}_{t,f,y}) \quad \forall t, y | y > y_0$$

$$\text{isrfi}_{t,y_0} = \text{uspare0}_t + \text{issue}_{t,y_0} + \sum_f (\text{tis}_{t,f,y_0} - \text{float}_{t,f,y_0}) \quad \forall t$$

- balances the pool of excess rfi assets

DOWNPOOL

$$\text{isnrfi}_{t,y} = \text{isnrfi}_{t,y-1} + \sum_f (\text{tiu}_{t,f,y} - \text{strn}_{t,f,y}) \quad \forall t, y | y > y_0$$

$$\text{isnrfi}_{t,y_0} = \text{dspare0}_t + \sum_f (\text{tiu}_{t,f,y_0} - \text{strn}_{t,f,y_0}) \quad \forall t$$

- balances the pool of excess nrfi assets

RFIBAL

$$\text{rfi}_{t,f,y} = \text{rfi}_{t,f,y-1} + \text{rpr}_{t,f,y} + \text{float}_{t,f,y} - \text{tis}_{t,f,y} - \text{usr}_{t,f,y} - \text{wo}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{rfi}_{t,f,y_0} = \text{srfi}_{t,f} + \text{cheat}_{t,f} + \text{rpr}_{t,f,y_0} + \text{float}_{t,f,y_0} - \text{tis}_{t,f,y_0} - \text{usr}_{t,f,y_0} - \text{wo}_{t,f,y_0} \quad \forall t, f$$

- establishes the balance of (non-excess) rfi assets

NRFIBAL

$$\text{nrfi}_{t,f,y} = \text{nrfi}_{t,f,y-1} - \text{rpr}_{t,f,y} - \text{tiu}_{t,f,y} + \text{strn}_{t,f,y} + \text{usr}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{nrfi}_{t,f,y_0} = \text{sbl}_{t,f} - \text{rpr}_{t,f,y_0} - \text{tiu}_{t,f,y_0} + \text{strn}_{t,f,y_0} + \text{usr}_{t,f,y_0} \quad \forall t, f$$

- establishes the balance of (non-excess) nrfi assets

UNFFBAL

$$\text{unfF}_{t,f,y} = \text{unfF}_{t,f,y-1} + \text{usr}_{t,f,y} - \text{rpr}_{t,f,y} \quad \forall t, f, y | y > y_0$$

$$\text{unfF}_{t,f,y_0} = \text{unfF0}_{t,f} + \text{usr}_{t,f,y_0} - \text{rpr}_{t,f,y_0} \quad \forall t, f$$

- establishes the balance of condition code F assets

RPRLIMIT

$$\text{rpr}_{t,f,y} \leq \text{usr}_{t,f,y} + (1 - \text{limRpr})\text{unfF}_{t,f,y-1} \quad \forall t, f, y | y > y_0$$

$$\text{rpr}_{t,f,y_0} \leq \text{usr}_{t,f,y_0} + (1 - \text{limRpr})\text{unfF0}_{t,f} \quad \forall t, f$$

- this ensures that only assets which have been returned or condition code F assets are available for repair

3 Architecture

This section discusses the general architecture of the ROME model.

3.1 Prerequisites

The following are required to run ROME in its current state:

- PC with Windows operating system
- Microsoft Excel
- GAMS (a version of ROME that uses an open-source alternative to GAMS called CMPL ((Coliop—Coin) Mathematical Programming Language) has also been developed)
- Python 3
- A username and password for MARCORLOGCOM Master Data Repository (MDR)
- An ODBC (Open Database Connectivity) connection to the MDR

3.2 Directory Structure

ROME relies on relative references to directories and files. It is important, therefore, to maintain the directory structure. The main directory (called `ROME20XXvY`) should contain the following subdirectories and files:

- `batchOutput`: contains the output from running a batch with different funding constraints
- `doc`: contains this manual, VV&A documentation and presentation materials
- `input`: contains the text input files for the GAMS script as well as the WBS Excel Spreadsheet
- `output`: contains the GAMS output as well as the SNaP csv output
- `scripts`: contains additional GAMS scripts
- `stratResults`: contains spreadsheets produced as output from the Strat Tool
- `dero.gms`: the main GAMS script
- `dracm.py`: the DRACM program to produce warfighting values (WFV)

- `genSnap.bat`: a batch file to run the SNaP-producing Python script
- `makeSnap.py`: Python script to create the SNaP exhibit
- `ROME.xlsx`: the main front-end for the ROME
- `runDERO.bat`: a batch file to run the GAMS script
- `runDRACM.bat`: a batch file to run the DRACM Python script

4 Preparing to Run the Model

Preparation for running the model consists of gathering all the required input and shaping it into the correct format. The SMEs in the Decision Support Branch of S&A division can be very helpful in tracking down the data. The ROME model aims to do much of the data shaping; however, it is often the case that the format that the data comes to the analyst will not be in the format that the ROME is expecting. The analyst may choose to format the data so that it is digestible by the ROME front-end, modify the ROME front-end to adapt to the new input format, or bypass the ROME front-end entirely and format the input directly into the GAMS input format. The endstate is that the data needs to be in the format that the GAMS script can digest. For examples, see the input text files in the `input` directory.

This step will likely take some work and creativity on the analyst's part. The path taken here will be highly dependent on the acquired knowledge and experience of the individual analyst. However, here is an example that may serve to get the "creative juices" flowing. Suppose that you are given the unservicable returns input in two separate Excel spreadsheets (one for active and one for reserves) with TAMCN as the first column, and the years as the remaining columns. You notice that to load this data into the GAMS program, each TAMCN/funding-pool/year combination must be a single row with the field names separated by a ".". Clearly, you are going to require a tool with the capability to input, parse, manipulate and output text. You will probably want to use the tool with which you are most familiar; suppose it is a Unix-like terminal shell. You might begin by researching how to input an Excel spreadsheet into some shell tool using the built-in documentation. Perhaps it is not possible to directly import an Excel spreadsheet but you discover that you can easily read in and work with a `.csv` file. You research how to convert an Excel file to `.csv` in your "Microsoft Excel 2010 Bible" and the process continues . . . Hopefully that helps.

Below are (somewhat inexplicably) a screenshot of the `input` directory and an example input file. If you need further help on GAMS input formatting, try typing "gams input format" into the internet search engine of your choice.

```
C:\Windows\system32\cmd.exe

Volume in drive C is SYSTEM
Volume Serial Number is EE58-D043

Directory of C:\Users\antonio.demarco\Documents\PPM\PPM_Development\ROME_Workin
gCopy\input

11/17/2013  04:40    <DIR>          .
11/17/2013  04:40    <DIR>          ..
07/16/2014  13:58                433 BUDGET.txt
07/16/2014  13:15                3,220 DSPARE0.txt
01/21/2014  11:29                 49 FORCESET.txt
07/16/2014  13:16            45,348 ICOST.txt
09/14/2012  16:03                 90 INTCPT.txt
07/16/2014  13:16            26,426 ISSUE.txt
07/16/2014  13:16           150,177 RPCT.txt
07/16/2014  13:15            18,714 SBL.txt
09/14/2012  14:29                 29 SEGMISET.txt
09/14/2012  16:02                 89 SLOPE.txt
07/16/2014  13:15            19,291 SRFI.txt
01/30/2013  14:20            2,487 TAMCNSET.txt
07/16/2014  13:15            18,614 UNFF0.txt
07/16/2014  13:15            3,107 USPARE0.txt
07/16/2014  13:28           136,095 USR.txt
07/16/2014  13:17            6,255 VALUE.txt
06/27/2013  14:30             8,132 WBS.csv
07/16/2014  13:15           140,901 WMR.txt
07/16/2014  13:16           135,325 WO.txt
04/09/2013  09:55                 48 YEARESET.txt
                20 File(s)          714,830 bytes
                2 Dir(s)  148,357,218,304 bytes free
```

```
C:\Windows\system32\cmd.exe

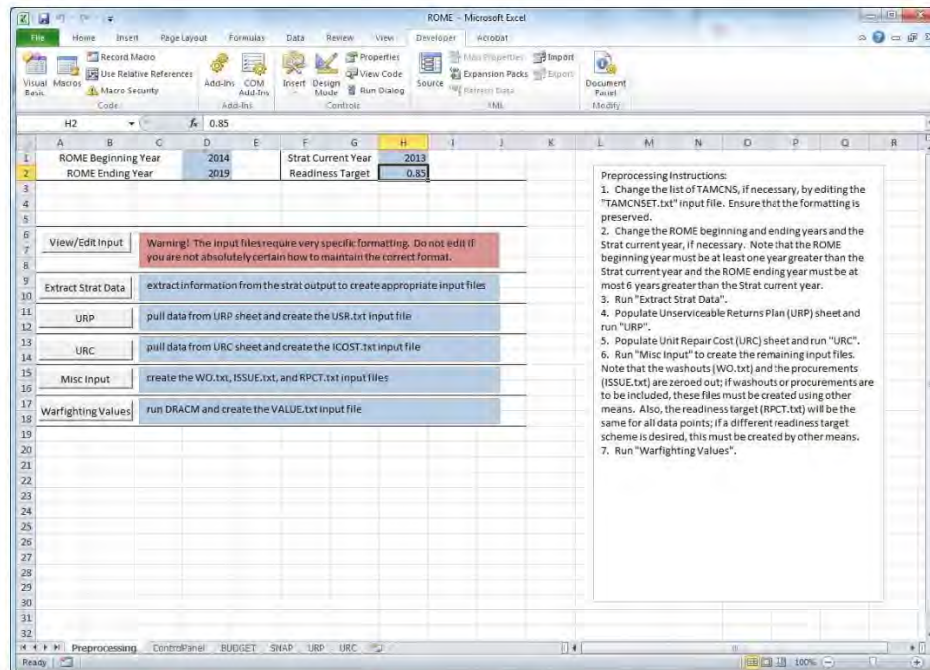
inputDir>type BUDGET.txt
parameter BUDGET(F,Y)
/
ACT_BASELINE.2014 100000000
ACT_BASELINE.2015 268879000
ACT_BASELINE.2016 171348000
ACT_BASELINE.2017 174037000
ACT_BASELINE.2018 175930000
ACT_BASELINE.2019 179097000
ACT_SUP.2014 570000000
ACT_SUP.2015 0
ACT_SUP.2016 0
ACT_SUP.2017 0
ACT_SUP.2018 0
ACT_SUP.2019 0
RES_BASELINE.2014 0
RES_BASELINE.2015 0
RES_BASELINE.2016 0
RES_BASELINE.2017 0
RES_BASELINE.2018 0
RES_BASELINE.2019 0
/
inputDir>
```

4.1 Inputs

Input Name	Type	Description	Source	Responsible Agency
$s \in S$	Index Set	indices for the lines which bound the $\text{score}_{t,f,y}$ variable used in the objective function	Manual Input	S&A Division OR Analyst
$t \in T$	Index Set	indices for the complete set of TAMCNs of interest	Manual Input	WSMC
$f \in F$	Index Set	indices for the funding pools of interest	Manual Input	WSMC
$y \in Y$	Index Set	indices for the years of interest	Manual Input	WSMC
$\text{wmr}_{t,f,y}$	Parameter	war materiel requirement for t from f during y	MDR	LOGCOM
$\text{usr}_{t,f,y}$	Parameter	unserviceable returns that are forecast to occur of t from f during y	Manual Input	WSMC
$\text{wo}_{t,f,y}$	Parameter	washouts that are forecast to occur of t from f during y	N/A	not currently used
$\text{rtgt}_{t,f,y}$	Parameter	readiness target of t for f during y	Manual Input	WSMC
$\text{icost}_{t,y}$	Parameter	IROAN cost per unit t during y	MDR	LOGCOM
$\text{issue}_{t,f}$	Parameter	number of t forecast to be issued during y from new procurement	N/A	not currently used
$\text{budget}_{f,y}$	Parameter	the budget for f during y	Manual Input	WSMC
$\text{srfi}_{t,f}$	Parameter	the starting number of stratified rfi t from f	MDR	LOGCOM
$\text{sbl}_{t,f}$	Parameter	the starting number of stratified nrfi t from f	MDR	LOGCOM
uspare0_t	Parameter	the starting number of non-stratified, rfi t	MDR	LOGCOM
dspare0_t	Parameter	the starting number of non-stratified, nrfi t	MDR	LOGCOM
value_t	Parameter	“warfighting value” of t	DRACM	LCC/S&A
slope_s	Parameter	slope of s	Manual Input	S&A Division OR Analyst
intercept_s	Parameter	intercept of s	Manual Input	S&A Division OR Analyst
discount	Parameter	fractional value of repairing an item in year y versus year $y - 1$	Manual Input	S&A Division OR Analyst
y_0	Parameter	the first year, taken as the earliest year in Y	Calculation Script	S&A Division OR Analyst
$\text{tilim}_{t,f,y}$	Parameter	the maximum number of t allowed to be turned in from f during y	Calculation Script	S&A Division OR Analyst

Input Name	Type	Description	Source	Responsible Agency
pen	Parameter	a penalty for being short of $rtgt_{t,f,y}$, for using $cheat_{t,f}$ variable or for being in excess of $wmr_{t,f,y}$	Calculation Script	S&A Division OR Analyst

4.2 Some Notes on Inputs



- The above screenshot might raise some questions such as “What is the correct format for the TAMCN input file?”, “What method could I use to create the input files for washouts or procurements?”, “How could I create an input file that reflects a different set of readiness targets?” and “There is a rather frightening warning (in red, with an exclamation point) about the correct input format. What is the correct format? I certainly don’t want to cause any harm.”, etc. If any of these or similar questions are weighing on your mind, first read the introduction to this section where GAMS input formatting is discussed. If that doesn’t help to inspire a solution to the problem you are wrestling with, perhaps an example of what **not** to do would help. Suppose you are told that in addition to the TAMCNs that were run last year, we also want to include {D0098, D8097, E1984}. What you wouldn’t want to do is the naive method of copying the above list using Ctrl-C, opening up the applicable .txt file by double clicking, and pasting the list at the end using Ctrl-V. A few things are wrong with that method.

1. The rest of the TAMCNs are given as 7 characters.

2. The delimiter is the EOL (End Of Line) character and not a comma (i.e. each TAMCN must be on a separate line without a comma in between).
3. Anything after the required “/” EOF (End Of File) character will not be read by GAMS (i.e., all the TAMCNs must be contained in /.../).

I hope that helps to point you toward the solution of the particular problem that you may face.

- Use the “Preprocessing” tab in the front-end to control the input shaping tasks.
- All the index sets must be manually changed by editing the GAMS text input files.
- The inputs in the above table which are indicated as manual inputs for which the analyst is responsible are also modified by directly editing the GAMS text input files.
- The inputs which are indicated as having MDR as the source are created by the ROME by clicking the “Extract Strat Data” button. This assumes that the Strat Data is in the `stratResults` folder in the expected format.
- The “Strat Current Year” on the “Preprocessing” tab is simply the fiscal year which was identified to the Strat Tool as the current year. It will normally be the actual current fiscal year unless the Strat was run some time ago.
- The “urp” and “urc” inputs are created by the ROME by clicking the respective buttons. This assumes that the spreadsheets have been imported into the ROME front-end in the expected format.
- Since “wo”, “issue” and “rpct” are currently not used or modified, the ROME creates dummy files for these inputs by clicking the “Misc Inputs” button. This assumes that the index sets have been created already.
- The warfighting values are created by the ROME by clicking the “Warfighting Values” button.
- The budget input can be modified on the “BUDGET” tab by directly changing the applicable cells within the Excel sheet, then clicking the “Save Budget” button which saves the data to the `BUDGET.txt` file. Alternatively, the user can choose to run in batch mode which automatically changes the budget based on the parameters input on the Control Panel tab (see the next section, “Running the Model”). The “Load Budget” button simply displays the current data saved in the `BUDGET.txt` file.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1					parameter	BUDGET(F,Y)							
2		Load Budget			/								
3		Save Budget			ACT_BASELINE.2014	100000000							
4					ACT_BASELINE.2015	268879000							
5					ACT_BASELINE.2016	171348000							
6					ACT_BASELINE.2017	174037000							
7					ACT_BASELINE.2018	175930000							
8					ACT_BASELINE.2019	179097000							
9					ACT_SUP.2014	570000000							
10					ACT_SUP.2015	0							
11					ACT_SUP.2016	0							
12					ACT_SUP.2017	0							
13					ACT_SUP.2018	0							
14					ACT_SUP.2019	0							
15					RES_BASELINE.2014	0							
16					RES_BASELINE.2015	0							
17					RES_BASELINE.2016	0							
18					RES_BASELINE.2017	0							
19					RES_BASELINE.2018	0							
20					RES_BASELINE.2019	0							
21					/								
22													
23													
24													
25													
26													

- The “Strat Results” which are referred to on the preprocessing tab are the Excel Workbooks which are created by the PEI Strat Tool. They are contained within the **StratResults** directory. Screenshots of the directory contents and an example worksheet are shown below.

```
C:\Windows\system32\cmd.exe
StratResults>dir /W
Volume in drive C is SYSTEM
Volume Serial Number is EE58-D043

Directory of C:\Users\antonio.denarco\Documents\PPM\PPM_Development\ROME_Workin
gCopy\StratResults

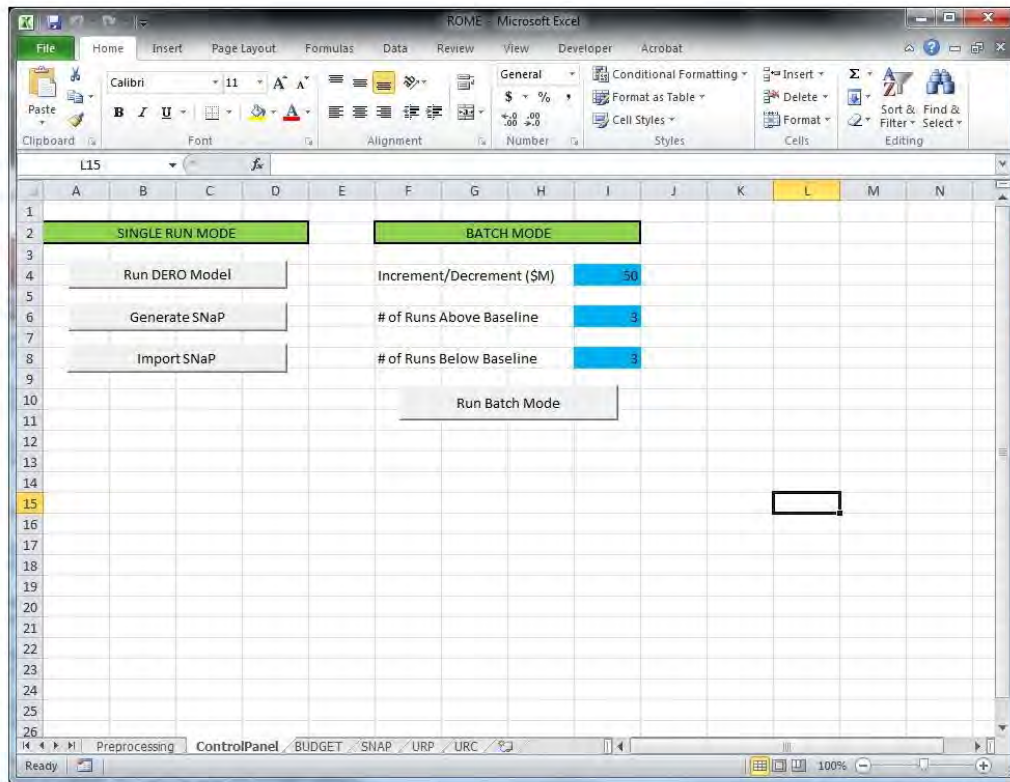
[.]                [..]                A00167G.xlsm      A00197G.xlsm      A00217G.xlsm
A00257G.xlsm      A00267G.xlsm      A00327G.xlsm      A00477G.xlsm      A00677G.xlsm
A00817G.xlsm      A01227G.xlsm      A01387G.xlsm      A01497G.xlsm      A01537G.xlsm
A01557G.xlsm      A01697G.xlsm      A02427G.xlsm      A02917G.xlsm      A03057G.xlsm
A08067G.xlsm      A08077G.xlsm      A08807G.xlsm      A12607G.xlsm      A13807G.xlsm
A14407G.xlsm      A15007G.xlsm      A15037G.xlsm      A18187G.xlsm      A19577G.xlsm
A20437G.xlsm      A21797G.xlsm      A22937G.xlsm      A23337G.xlsm      A23352B.xlsm
A23362B.xlsm      A23372B.xlsm      A23382B.xlsm      A25257G.xlsm      A26007G.xlsm
A23227G.xlsm      A27007G.xlsm      A70017G.xlsm      A70027G.xlsm      A70057G.xlsm
A70097G.xlsm      A70257G.xlsm      A70407G.xlsm      A70467G.xlsm      A70557G.xlsm
A70577G.xlsm      A70597G.xlsm      A70617G.xlsm      A70807G.xlsm      A70817G.xlsm
A70837G.xlsm      A70847G.xlsm      A70867G.xlsm      A70877G.xlsm      A75017G.xlsm
A75117G.xlsm      A75177G.xlsm      A75967G.xlsm      A75977G.xlsm      A80237G.xlsm
A80327G.xlsm      A80367G.xlsm      A80387G.xlsm      A80477G.xlsm      A80727G.xlsm
A80847G.xlsm      A80887G.xlsm      A80917G.xlsm      B00247B.xlsm      B00357K.xlsm
B00387B.xlsm      B00407B.xlsm      B00447B.xlsm      B00587B.xlsm      B00607B.xlsm
B00637B.xlsm      B00667B.xlsm      B00717B.xlsm      B00767B.xlsm      B00787B.xlsm
B00797B.xlsm      B00817B.xlsm      B01017B.xlsm      B01022B.xlsm      B01052B.xlsm
B01147B.xlsm      B01157B.xlsm      B01207B.xlsm      B01217B.xlsm      B01227B.xlsm
B01527B.xlsm      B01607B.xlsm      B01717B.xlsm      B02157B.xlsm      B03657B.xlsm
B03927B.xlsm      B03957B.xlsm      B04467B.xlsm      B04712E.xlsm      B05897B.xlsm
B06267B.xlsm      B06477B.xlsm      B07307B.xlsm      B08917B.xlsm      B09537B.xlsm
B10167B.xlsm      B10217B.xlsm      B10457B.xlsm      B11397B.xlsm      B12987B.xlsm
B13007B.xlsm      B13157B.xlsm      B14202B.xlsm      B15807B.xlsm      B15817B.xlsm
B15837B.xlsm      B17207B.xlsm      B17857B.xlsm      B20857B.xlsm      B20867B.xlsm
B23947B.xlsm      B24607B.xlsm      B24647B.xlsm      B24837B.xlsm      B25617B.xlsm
B25657K.xlsm      B25667B.xlsm      B26057B.xlsm      C00012G.xlsm      C00022G.xlsm
C00032G.xlsm      C00187B.xlsm      C01727B.xlsm      C21042E.xlsm      C22782E.xlsm
C22862E.xlsm      C22882E.xlsm      C45497B.xlsm      C47657B.xlsm      C52692E.xlsm
C70017B.xlsm      C70077B.xlsm      C70337B.xlsm      C79042B.xlsm      C79052B.xlsm
C79082B.xlsm      C79127B.xlsm      C79307B.xlsm      C80108B.xlsm      D00027K.xlsm
D00037K.xlsm      D00057K.xlsm      D00077K.xlsm      D00097K.xlsm      D00117K.xlsm
D00137K.xlsm      D00157K.xlsm      D00227K.xlsm      D00237K.xlsm      D00257K.xlsm
D00277K.xlsm      D00307K.xlsm      D00317K.xlsm      D00327K.xlsm      D00337K.xlsm
D00347K.xlsm      D00357K.xlsm      D00367K.xlsm      D00407K.xlsm      D00807K.xlsm
D01987K.xlsm      D02157K.xlsm      D02357K.xlsm      D08607K.xlsm      D08617K.xlsm
D08807K.xlsm      D08867K.xlsm      D08877K.xlsm      D10017K.xlsm      D10027K.xlsm
D10627K.xlsm      D10637K.xlsm      D10647K.xlsm      D11257K.xlsm      D11587K.xlsm
D11617K.xlsm      D11627K.xlsm      D12137K.xlsm      D12147K.xlsm      E00137B.xlsm
E00177M.xlsm      E00192E.xlsm      E00422B.xlsm      E00547G.xlsm      E00557M.xlsm
E00562E.xlsm      E00587G.xlsm      E00602B.xlsm      E00627G.xlsm      E00637G.xlsm
E00647G.xlsm      E00667G.xlsm      E00697B.xlsm      E00707B.xlsm      E00767B.xlsm
E00827B.xlsm      E00862E.xlsm      E00887B.xlsm      E00907G.xlsm      E00927G.xlsm
E01237M.xlsm      E01337G.xlsm      E01497B.xlsm      E01507B.xlsm      E01807B.xlsm
E01907M.xlsm      E01957M.xlsm      E02057M.xlsm      E02077B.xlsm      E03307M.xlsm
E03602E.xlsm      E03902E.xlsm      E06717M.xlsm      E07967K.xlsm      E08467K.xlsm
E08567K.xlsm      E08927M.xlsm      E09157M.xlsm      E09307B.xlsm      E09357M.xlsm
E09427M.xlsm      E09467B.xlsm      E09477M.xlsm      E09487B.xlsm      E09497M.xlsm
E09507B.xlsm      E09607M.xlsm      E09807M.xlsm      E09847M.xlsm      E09897M.xlsm
E09947M.xlsm      E09967M.xlsm      E09977M.xlsm      E09987M.xlsm      E09997M.xlsm
E10657M.xlsm      E10957M.xlsm      E11607G.xlsm      E13787K.xlsm      E14422M.xlsm
E14602M.xlsm      E15007M.xlsm      E17107G.xlsm      E17797B.xlsm      E17987G.xlsm
E18342E.xlsm      E18397M.xlsm      E18887M.xlsm      E19117B.xlsm      E19757G.xlsm
E19767G.xlsm      E19787G.xlsm      E31967B.xlsm      U00137B.xlsm      U00147B.xlsm
U30207B.xlsm

270 File(s)          27,633,474 bytes
2 Dir(s)             141,291,270,144 bytes free
```


5 Running the Model

Needless to say, the difficult step of gathering and shaping the data must be completed prior to running the model. Running the model is the easy part. The controls on the “Control Panel” tab may or may not be self-explanatory. The model can be run with a single budget input or in batch mode with automated changes to the budget. In batch mode, a SNaP is not produced for each run but the GAMS output for each run is saved individually in the **batchOutput** directory. To run in batch mode, a baseline budget (this is usually the expected budget) is input using the method described at the end of Section 4 of this document. On the Control Panel tab, the increment/decrement amount and number of runs above and below baseline are input into the appropriate cells. For example, in the below screenshot, a total of 7 runs would occur; one at baseline, one at \$50M above baseline, one at \$100M above baseline, one at \$150M above baseline, one at \$50M below baseline, one at \$100M below baseline, one at \$150M below baseline. These are usually representative of a range of possible alternative budgets that may be imposed. It is not the prerogative of the analyst to choose the dollar amounts, number above and below baseline, etc. These are inputs which are given by the appropriate stakeholders.

As far as the order of “button pressing” in the single run mode; first the DERO optimization is run. This may be the only step unless a SNaP exhibit is required. If a SNaP is required in addition to the raw GAMS output, the SNaP is first generated then imported using the appropriate buttons. After pressing any of these buttons, a shell containing the running process will pop up. In general, the DERO model and the Extract Strat processes should take several minutes to complete, and all the other processes on the Control Panel or the Preprocessing tabs should take less than one minute.



Below are screenshots of the SNAP, URP and URC tabs. The SNAP tab contains the imported SNaP exhibit. The URP and URC tabs contain the Unservicable Returns Plan and Unit Repair Cost inputs respectively.

JL WeaponsSystem																										
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y		
1	Class	Comd	Depot	Method	Of	Mainten	Mainten	Maint	Weapon	System	WorkPc	PRICE	Treas	BACD	BUDGE	Program	Resource	Type	Comme	FY2014	FY2015	FY2016	FY2017	FY2018	FY2019	
2	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	8003	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	Units Required	8	7	7	14	13	7					
3	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	Units Funded	8	7	7	14	13	7						
4	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	Units Required Supple	2	1	12	0	0	0	0	0	0	0		
5	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	Units Funded Supple	2	0	0	0	0	0	0	0	0	0		
6	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	TOA Required	3054992	2707889	2753902	5623469	5331450	2931067						
7	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	TOA Direct Funded	3054992	2707889	2753902	5623469	5331450	2931067						
8	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	TOA Required Supple	763748	386816	4720975	0	0	0	0	0	0	0		
9	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRUCK,BOMB SEI	1106	01	1A-1A3	0702207	TOA Funded Supple	763748	0	0	0	0	0	0	0	0	0		
10	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	8009	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	Units Required	0	0	9	9	9	9	9	9			
11	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	Units Funded	0	0	0	0	0	0	0	0	0	0		
12	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	Units Required Supple	0	0	0	0	0	0	0	0	0	0		
13	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	Units Funded Supple	0	0	0	0	0	0	0	0	0	0		
14	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	TOA Required	0	0	311306	317845	324520	331335						
15	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	TOA Direct Funded	0	0	0	0	0	0	220890					
16	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	TOA Required Supple	0	0	0	0	0	0	0	0	0	0		
17	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	MINE,ROLLER SNI	1106	01	1A-1A3	0702207	TOA Funded Supple	0	0	0	0	0	0	0	0	0	0		
18	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	8011	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	Units Required	9	0	0	0	0	0	0	0	0		
19	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	Units Funded	9	0	0	0	0	0	0	0	0	0		
20	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	Units Required Supple	0	0	0	0	0	0	0	0	0	0		
21	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	Units Funded Supple	0	0	0	0	0	0	0	0	0	0		
22	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	TOA Required	1604137	0	0	0	0	0	0	0	0	0		
23	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	TOA Direct Funded	1604137	0	0	0	0	0	0	0	0	0		
24	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	TOA Required Supple	0	0	0	0	0	0	0	0	0	0		
25	U	Active	Marine	Consumable	N/A	Automot	Support	V Other	BOAT,BRIDGE EFI	1106	01	1A-1A3	0702207	TOA Funded Supple	0	0	0	0	0	0	0	0	0	0		
26	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	8011	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	Units Required	8	0	0	0	0	0	0	0	0		
27	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	Units Funded	8	0	0	0	0	0	0	0	0	0		
28	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	Units Required Supple	0	0	0	0	0	0	0	0	0	0		
29	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	Units Funded Supple	0	0	0	0	0	0	0	0	0	0		
30	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	TOA Required	69968	0	0	0	0	0	0	0	0	0		
31	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	TOA Direct Funded	69968	0	0	0	0	0	0	0	0	0		
32	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	TOA Required Supple	0	0	0	0	0	0	0	0	0	0		
33	U	Active	Marine	Maintenano	Organic	Automot	Support	V Other	TRAILER,BRIDGE	1106	01	1A-1A3	0702207	TOA Funded Supple	0	0	0	0	0	0	0	0	0	0		
34	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	8012	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	Units Required	0	0	0	0	1	1				
35	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	Units Funded	0	0	0	0	1	1					
36	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	Units Required Supple	0	0	2	0	0	0	0	0	0		
37	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	Units Funded Supple	0	0	0	0	0	0	0	0	0		
38	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	TOA Required	0	0	0	0	459615	469267					
39	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	TOA Direct Funded	0	0	0	0	459615	469267					
40	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	TOA Required Supple	0	0	881895	0	0	0	0	0	0		
41	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE ELECTIC	1106	01	1A-1A3	0702207	TOA Funded Supple	0	0	0	0	0	0	0	0	0		
42	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	8013	BRIDGE, FIXED (I)	1106	01	1A-1A3	0702207	Units Required	0	2	1	2	1	1				
43	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE, FIXED (I)	1106	01	1A-1A3	0702207	Units Funded	0	2	1	0	0	0	0	0	0		
44	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE, FIXED (I)	1106	01	1A-1A3	0702207	Units Required Supple	1	2	3	0	0	0	0	0	0		
45	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE, FIXED (I)	1106	01	1A-1A3	0702207	Units Funded Supple	1	0	0	0	0	0	0	0	0		
46	U	Active	Marine	Anniston	An	Organic	Automot	Support	V Other	BRIDGE, FIXED (I)	1106	01	1A-1A3	0702207	TOA Required	0	1300321	661211	1356198	689276	703751					

TAMCN	13 Cost	14 Cost	15 Cost	16 Cost	17 Cost	18 Cost	19 Cost
A0016G	1,353,089.00	1,376,091.51	1,400,861.26	1,426,076.66	1,451,746.04	1,477,877.47	1,504,479.26
A0017G	38,670.41	41,648.05	42,189.45	42,506.67	43,807.71	44,727.66	45,666.96
A0027G	79,836.00	81,199.21	82,854.69	84,142.47	85,657.04	87,196.87	88,768.45
A0028G	71,723.93	72,943.23	74,256.21	75,592.82	76,953.49	78,338.66	79,748.75
A0029G	2,068.58	2,227.86	2,256.82	2,295.19	2,343.39	2,392.60	2,442.84
A0032G	530,071.60	539,082.82	548,786.31	558,664.46	568,720.42	578,957.39	589,378.62
A0047G	299,650.00	299,650.00	299,650.00	299,650.00	299,650.00	299,650.00	299,650.00
A0057G	34,602.10	34,602.10	34,602.10	34,602.10	34,602.10	34,602.10	34,602.10
A0059G	410.15	410.15	410.15	410.15	410.15	410.15	410.15
A0112G	323,679.59	329,182.54	335,107.83	341,139.77	347,280.28	353,531.33	359,894.89
A0138G	26,000.00	26,000.00	26,000.00	26,000.00	26,000.00	26,000.00	26,000.00
A0149G	303,075.15	308,227.43	313,775.52	319,423.48	325,173.11	331,025.22	336,984.69
A0153G	146,145.35	146,145.35	146,145.35	146,145.35	146,145.35	146,145.35	146,145.35
A0155G	34,107.48	34,687.31	35,311.68	35,947.29	36,594.34	37,253.04	37,923.59
A0169G	221,704.60	221,704.60	221,704.60	221,704.60	221,704.60	221,704.60	221,704.60
A0240G	191,750.00	191,750.00	191,750.00	191,750.00	191,750.00	191,750.00	191,750.00
A0305G	1,727,050.00	1,727,050.00	1,727,050.00	1,727,050.00	1,727,050.00	1,727,050.00	1,727,050.00
A0806G	290,400.00	295,336.80	300,852.86	306,064.61	311,573.78	317,182.10	322,891.38
A0807G	290,400.00	295,336.80	300,852.86	306,064.61	311,573.78	317,182.10	322,891.38
A0880G	78,843.05	78,843.05	78,843.05	78,843.05	78,843.05	78,843.05	78,843.05
A1268G	482.39	490.59	499.42	508.41	517.57	526.88	536.37
A1308G	221,397.85	225,161.61	229,224.52	233,440.38	237,840.51	241,816.24	246,168.93
A1401G	1,271,250.00	1,292,861.25	1,316,152.75	1,341,833.14	1,369,393.96	1,398,490.88	1,413,483.71
A1508G	1,353,368.68	1,376,375.95	1,401,130.72	1,426,371.43	1,452,046.12	1,478,182.95	1,504,790.24
A1503G	2,542,500.00	2,585,722.50	2,632,265.51	2,679,646.28	2,727,879.92	2,776,981.76	2,826,967.43
A1818G	19,570.13	21,077.03	21,351.03	21,714.00	22,189.99	22,635.56	23,110.91
A1957G	28,590.90	28,590.90	28,590.90	28,590.90	28,590.90	28,590.90	28,590.90
A2043G	5,240.30	5,240.30	5,240.30	5,240.30	5,240.30	5,240.30	5,240.30
A2179G	290,315.90	313,116.43	317,889.54	322,785.16	329,263.05	336,484.49	343,550.66
A2333G	487,156.92	475,095.38	489,850.36	492,156.06	501,218.47	510,340.41	519,424.73
A2331G	172,400.14	175,330.94	178,486.90	181,699.67	184,970.26	188,299.72	191,689.12
A2352B	66,802.66	71,946.47	72,881.77	74,120.76	75,677.30	77,266.52	78,889.12
A2336B	74,910.19	80,678.27	81,727.09	83,116.45	84,861.89	86,643.99	88,463.52
A2337B	98,365.26	105,939.38	107,316.59	109,140.98	111,432.94	113,773.03	116,162.26
A2382B	35,925.53	38,691.79	39,194.78	39,861.10	40,698.12	41,552.84	42,425.45
A2528G	278,287.81	285,018.70	288,113.04	293,209.06	298,576.46	303,952.87	309,424.02
A2698G	1,430,000.00	1,430,000.00	1,430,000.00	1,430,000.00	1,430,000.00	1,430,000.00	1,430,000.00
A2323G	176,958.00	179,966.29	183,205.88	186,503.38	189,880.44	193,277.93	196,756.93
A2707G	69,811.97	70,998.77	72,276.75	73,577.73	74,902.13	76,250.37	77,622.87
A2708G	864.40	879.09	894.52	911.03	927.43	944.12	961.11
A2702G	2,394.02	2,578.36	2,611.88	2,656.28	2,712.06	2,769.01	2,827.16
A2709G	9,575.06	10,312.33	10,446.39	10,623.98	10,847.09	11,074.88	11,307.45
A2700G	4,748.99	5,115.09	5,181.59	5,289.68	5,380.34	5,493.31	5,600.69
A2705G	5,538.46	3,810.92	3,860.48	3,928.09	4,008.34	4,092.72	4,176.67
A2740G	27,773.46	28,245.61	28,754.03	29,271.60	29,798.49	30,334.86	30,880.89

5.1 Output

The GAMS output consists simply of all of the parameter and optimal variable values for each of the TAMCN/funding-pool/year combinations. Experience shows that this output, with some irrelevant columns removed and with a description of each of the relevant columns, is sufficient for decision makers to digest. The SNaP exhibit is produced to satisfy a reporting requirement associated with the ELMP process. A thorough description of the SNaP can be found at <https://snap.cape.osd.mil>.

It should be noted that the output costs are in “then-year” dollars. This is a consequence of the fact that the repair costs and budget are given as an input in then-year dollars. Since monies are not carried over year-to-year in the model, if the input costs and budget are given in current-year dollars, then the output would be in current-year dollars as well.

6 Capabilities, Limitations and Assumptions

6.1 Capabilities

The ROME is designed to support the following objectives:

- Help lifecycle asset managers satisfy the many competing objectives (e.g., affordability, operational availability, maintenance line availability, and the need to increase Marine Corps readiness ratings) to find the proper mix of equipment to induct into depot maintenance.
- Adapt to changing circumstances. The tool allows users to alter model input parameters and constraints to appropriately reflect changing circumstances (e.g., budget increases or decreases, changing equipment priorities, changes in availability of equipment).
- Enable planners and analysts to compare multiple scenarios and assess the impacts of different assumption sets (i.e., different model input parameters and constraints) against multiple competing priorities (e.g., affordability, operational availability, 50/50 and Core).
- Enable rapid setup, execution of model runs, and analysis of model output in support of rapid what-if analyses.
- Assist lifecycle asset managers with producing repeatable, auditable, quantitative input to Program Objective Memorandum (POM) and budget submissions.

6.2 Limitations

The ROME is limited by accepted organizational constraints known as business rules. Business rules are mathematically stated in Section 2 of this document. The model is currently limited to one user due to the GAMS license and user documentation/training (although a version using an open-source alternative to GAMS called CMLP has been developed).

6.3 Assumptions

In addition to the assumptions of a mixed-integer linear program (which may be found in any introductory text on optimization, or at http://en.wikipedia.org/wiki/Linear_programming), there are many assumptions which impact the formulation of the business rules in Section 2.

- The modeling and formulation of the current business rules (circa Dec 2010) are incorporated to the extent practicable (e.g., the separation of the term force to represent both the funding sources as well as the units owning the assets). As the ELMP PPM process is further refined and business rules change, the formulation will require updating.
- There is an implicit assumption in this formulation that all planned returns of NRFI assets, noted as unserviceable returns (USR), are not only returned, but are returned in sufficient time and quantity to preclude any negative impact to the repair line. The current USMC methodology requires the forces to return the NRFI assets to the repair depots no later than 30 days prior to the quarter for which the induction is scheduled.
- In the event that planned NRFI quantities are not returned, delays or deletion of induction of planned repair items would result. In this event, the quantities planned using the PPM would not be repaired, leading to an overall departure from the PPM best solution. Funds planned for the deleted items would be returned in the worst case scenario, or reprogrammed to another Master Work Schedule line item instead.
- There is an implicit assumption that funded repairs actually get repaired in the year they are planned and funding is approved. While this is the usual procedure, real-world events and other considerations (such as the non-arrival of items for induction) often result in depot repair schedule changes.
- All new procurement items are generally sent directly to the operational units from the Original Equipment Manufacturer (OEM), vendor, or source of material and have no effect upon the planning or execution process.
- All Defense Logistics Agency / Disposition Services (DLA/DS) actions, such as system losses, occur for the in-stores inventory at the depot level. These annual losses, or wash outs, are provided in an input file and are based on historical data.
- All NRFI assets which meet depot maintenance evacuation criteria are either returned to RFI status within 30 days by repairs made at the unit level or they are returned to the depot inventory and placed in a NRFI status.
- Assets assigned materiel condition codes of A-C are categorized as RFI.
- Assets with materiel condition codes of H, P, R, or S are categorized as washouts.
- Assets assigned materiel condition codes other than A-C, H, P, R, or S, are categorized as NRFI.
- only those assets assigned condition codes D-F will be considered candidates for depot maintenance.
- Item Managers assign materiel condition codes correctly.
- All planned Unserviceable Returns (USR) are assigned a condition code of F.

- Warfighting values produced using the Dimension Reduction and Clustering Means (DRACM) Model will be used until an enterprise level capability to identify qualifying TAMCNs is developed. VV&A for the DRACM model is complete.
- Principle End Item (PEI) Stratification is performed and accurately forecasts the future use of equipment in the Marine Corps.

7 Configuration Management

If a request to alter the ROME model is made, the Studies and Analysis Division will evaluate the proposed request and submit to the current licensed GAMS user (the analyst) to alter the code. The request to alter the model must include a description of the error that the model has, evidence for the error and a proposed solution to the problem. Careful consideration of the effect of any code change to the operation of the code will be made. An alpha version of the code will be produced and ran with test data to determine if any errors are produced. If the code passes this test another trial will be ran with data where the results are known to test the program again. Documentation of changes will be made as comments within the code itself and the main directory will be renamed in a sequential manner with the year and version (e.g. `ROME2014v3`). Updates to this document will be made to reflect any code changes. Since there is only a single user at any given time, version control is a non-issue.

APPENDIX B. ROMEFARMER USER’S MANUAL

ROMEfarmer User’s Manual, version 0.3 - DRAFT

Stephen C. Upton, SEED Center for Data Farming

2018-11-14

1. 1 Background

ROMEfarmer was developed as a part of the Naval Postgraduate School’s Naval Research Program project NPS-18-M319-A, “Improving Marine Corps Logistics with Model-driven Big Data,” in conjunction with the U. S. Marine Corps Logistics Command (MARCORMARCORLOGCOM).

2. 2 Overview

ROMEfarmer consists of three components: ROMEfarmer, ROMERunner, and ROMEminer. ROMEfarmer is the top-level interface to the user and is responsible for executing the designed experiment. ROMERunner is the component that takes one set of inputs, also called a design point, case or excursion, from a designed experiment and runs the ROME model for that instance. ROMEminer takes the output from the designed experiment and post-processes that output by concatenating the output files into a single output file for further analysis.

3. 3 Installing ROMEfarmer

4. 3.1 Prerequisites

The following software needs to be installed before running ROMEfarmer. We assume the user is on a Windows computer, although there is no inherent reason why ROMEfarmer, and ROME, could not be run on a linux machine except for the use of Excel for the front-end GUI, since both GAMS and python can run on linux.

1. GAMS (we are currently using version 23.8 to be compatible with MARCORMARCORLOGCOM use)
2. python, version 3.6 (see “Installing python” in the Appendix)

3. Microsoft Excel, (we are currently using Excel 2016)

GAMS is required since that is a pre-requisite to run the ROME model. ROMEfarmer is written in python 3.6. If you use a basic python installation, ROMEfarmer also requires the numpy and pandas packages (which require a separate download). Excel is used for ROMEfarmer's (bare-bones, prototypical) front-end graphical user interface (GUI).

4. 3.2 Installing ROMEfarmer

ROMEfarmer is provided as a zipped file. Installation is completed by simply unzipping the file, ROMEfarmer.zip, to a suitable location on your computer (preferably using a path that does not contain spaces, e.g., C:\ROMEfarmer).

After unzipping ROMEfarmer, your ROMEfarmer folder should have the following structure:

- docs
 - ROMEfarmerUserManual.docx (this document)
- DOE
 - NOLHdesigns_v6.xls
- source (includes all the python source code)
 - InputModifier.py
 - ROMEfarmer.py
 - ROMErunner.py
 - ROMEminer.py
- studies
 - sample_study
 - sampledoe.csv (a small design used for this example)
- ROMEfarmer-frontend.xlsm - this is an Excel (bare-bones) front-end GUI
- ROMEmodel.zip

In the discussion below, we will assume that the user has installed (unzipped) ROMEfarmer to "c:\ROMEfarmer".

5. 4 Data farming process

This section describes the overall process to data farm the Repair Optimization Materiel Evaluator (ROME) model with ROMEfarmer.

In brief, the data farming process using ROMEfarmer is as follows:

6. Construct the set of inputs for ROME
7. Identify the factors and ranges of interest and create your design of experiments file
8. Run ROME over the set of design points with ROMEfarmer
9. Post-process and analyze the output
10. Repeat as desired

The general process is depicted in Figure 4.1, Data Farming Process.

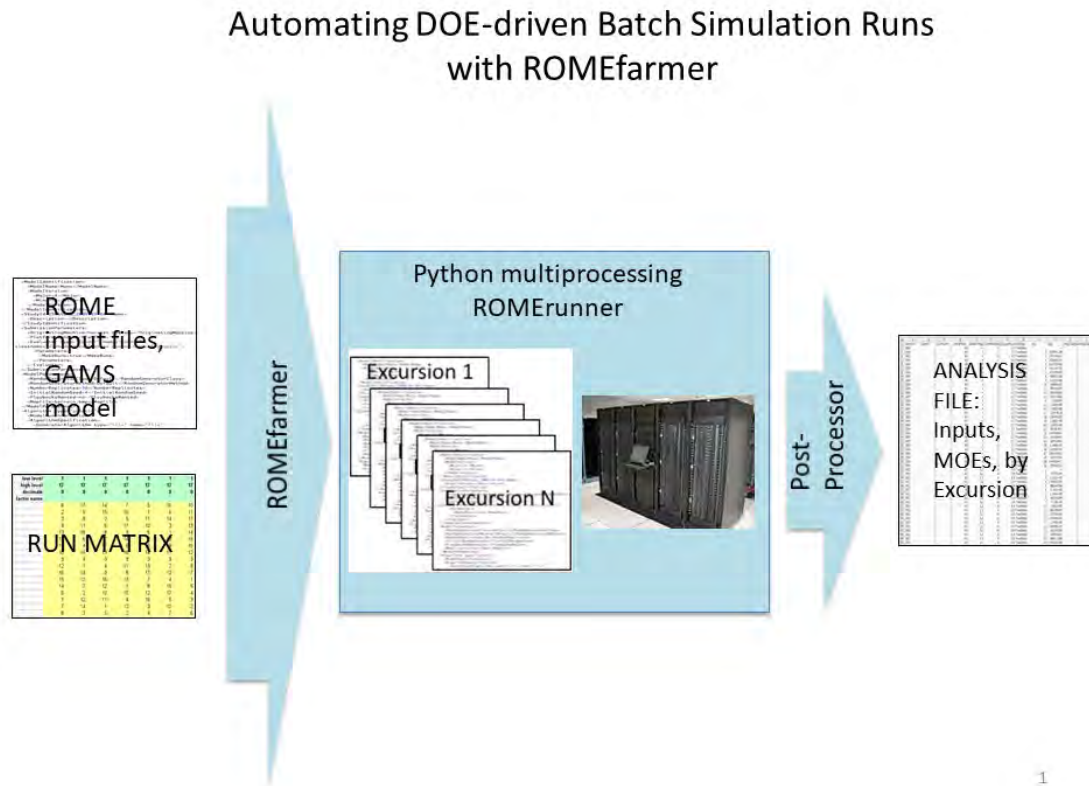


Figure 4.1: ROMEfarmer Data Farming Process

Below, we will describe these steps in more detail. We will illustrate the process by walking through a simple example using a small design and the set of input files from the ROMEfarmer distribution (which were provided by MARCORMARCORLOGCOM).

1. Construct the set of inputs for ROME

This step is currently outside the scope of ROMEfarmer and is a pre-requisite to running ROME and ROMEfarmer. The analyst must first construct the set of inputs that are needed for the ROME/GAMS model, which are then modified by ROMEfarmer for each design point. The required input files are listed in the `dero.gms` file. ROMEfarmer expects these to be in a particular structure (which

mimics how the dero.gms file expects them - as currently written): the text files are located in a folder called “input” and the additional dero.gms files are expected to be in a folder called “scripts” (this currently includes a “calcddata.gms” file and a “makeOutput.gms” file). Additionally, there is a “runDERO.bat” file that is a Windows DOS batch file that includes the path to the local GAMS installation and a call to GAMS to run the ROME model [NB: we did add the logoption to the gams call ‘logoption 2’ so that a GAMS log file is also outputted; ROMEMiner uses this file to extract the objective value]. Once these files are collected and placed in the aforementioned structure, the analyst then zips the files into a single compressed file. This zip file is then one input to ROMEfarmer - you can name the file anything you like; the ROMEfarmer installation calls this “ROMEmodel.zip.” See the Appendix for more details on the structure of this zip file, which includes the list of required ROME input files.

For our simple example, a ROMEmodel.zip file is provided as part of the ROMEfarmer distribution.

2. Identify the factors and ranges of interest and create your design of experiments file

Our goal in this section is limited to describing how to construct the design of experiments (DOE) file used when data farming ROME. We assume the reader is somewhat familiar with designed experiments (for details on designed experiments, see e.g., the papers and presentations page of [SEED Center for Data Farming website](#)).

Briefly, we will do the following in this step:

1. Identify the factors
2. Identify their ranges
3. Construct the design using the SEED Center’s NOLH spreadsheet
4. Copy the header and the values to a new worksheet
5. Save the worksheet as a CSV file

The first step in constructing your DOE file is to identify the factors or variables that you want to change in your experiment. For our sample experiment, we will focus on a small set of factors using parameters associated with a single Principal End Item (PEI) (we chose E0942: LAV, Anti-Tank, but this is for illustrative purposes only; additional PEIs, as well as any variable or parameter in the ROME model is a potential factor, with the exception of the decision variables), along with a Budget, as an example:

- budget for the year 2019

- IROAN cost per unit of E0942 in 2018
- war materiel requirement for E0942 during 2019
- warfighting value of E0942

The second step is determining the range of values for each of the factors. For our example, we are interested in doing a sensitivity analysis of the parameters, so we will start with the original input of the above factors and adjust by plus/minus 15%. This is just as an example; other percentages could be used for the sensitivity analysis. Additionally, broad parameter sweeps, and other types of designed experiments are possible and useful. See Table 4.1 below for the original, minimum, and maximum values for each of the factors. We will use the minimum and maximum values to construct the design.

Table 4.1: Min/Max Ranges of the Factors

factor	original	Min	Max
BUDGET/ALL.2019	35665320	30315522	41015118
ICOST/E09427M.2018	550623.8	468030.3	633217.4
WMR/E09427M.2019	83	71	95
VALUE/E09427M	6.45027	5.48273	7.41781

Once we identified the parameters and their ranges of interest, we use the SEED Center for Data Farming’s NOLHdesigns_v6.xls spreadsheet to construct the DOE file. (see [SEED Center for Data Farming software downloads](#) under the “Experimental Design Tools” section; the spreadsheet is also included as part of the ROMEfarmer distribution under the DOE folder.) The “NOLH” in the file name is an acronym for Nearly Orthogonal Latin Hypercube, a space-filling design type that is one of the NPS SEED Center’s signature designs. See Figure 4.2 displays the “OLH for up to 7 factors” sheet. Other sheets contain larger designs (up to more factors, with more design points). See the workbook’s “readme” sheet for more details.

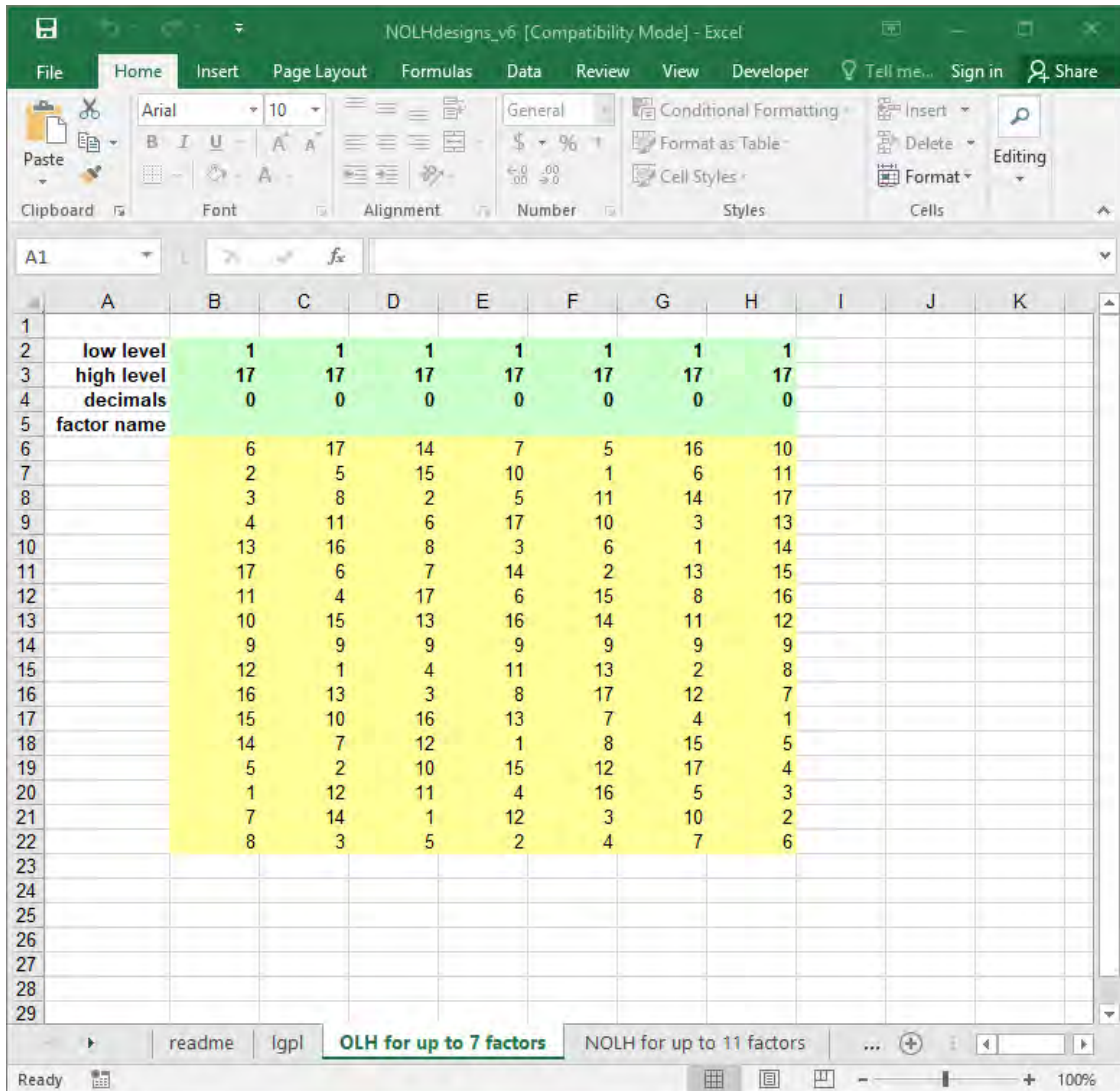


Figure 4.2: NOLH 7 factors

The DOE file will contain one column for each factor and one row for each design point. To create the design, the user simply fills out the header rows on the appropriate tab (depending on the number of factors and/or the desired number of design points - we will use the “OLH for up to 7 factors” sheet) with the column name and lower and upper limits for each factor. We will use the minimum and maximum values we constructed above. The decimal column is used to indicate how many decimal places you want to include in the value. Use 0 for integer values, i.e., where the factor values must be whole numbers.

In the header row, the DOE file factor names are constructed such that the ROME input file is specified, followed by a “/,” then by the desired variable name in that file (this should be one line/entry in the respective file). For example, if you want to

include the BUDGET for 2019, then the factor name would be “BUDGET/
ALL.2019.” This is the convention ROMEfarmer uses in order to find the value to
change in the input file for each design point, for each factor. See Figure 4.3 for a
completed design.

	A	B	C	D	E	F	G	H
1								
2	low level	30315522	468030.3	71	5.48273	1	1	1
3	high level	41015118	633217.4	95	7.417811	17	17	17
4	decimals	0	1	0	5	0	0	0
5	factor name	BUDGET/ALL.2019	ICOST/E09427M.2018	WMR/E09427M.ALL.2019	VALUE/E09427M			
6		33659146	633217.4	91	6.20839	5	16	10
7		30984247	509327.1	92	6.57121	1	6	11
8		31652972	540299.7	73	5.9665	11	14	17
9		32321696	571272.2	79	7.41781	10	3	13
10		38340219	622893.2	82	5.72462	6	1	14
11		41015118	519651.3	80	7.05498	2	13	15
12		37002770	499002.9	95	6.08744	15	8	16
13		36334045	612569	89	7.29687	14	11	12
14		35665320	550623.9	83	6.45027	9	9	9
15		37671494	468030.3	76	6.69216	13	2	8
16		40346393	591920.6	74	6.32933	17	12	7
17		39677669	560948	94	6.93404	7	4	1
18		39008944	529975.5	88	5.48273	8	15	5
19		32990421	478354.5	85	7.17593	12	17	4
20		30315522	581596.4	86	5.84556	16	5	3
21		34327871	602244.8	71	6.8131	3	10	2
22		34996595	488678.7	77	5.60367	4	7	6
23								
24								
25								
26								
27								
28								
29								

Figure 4.3: NOLH 7 factors with values

Once you have completed the design, select the appropriate header and the rows (B5 to E22 in Figure 4.3), and copy the values to another, blank sheet in the workbook, saving only the values. We then save this sheet to a CSV file as our completed design file that can be used by ROMEfarmer. We will name our file sampledoe.csv. See Table 4.2 for the completed design.

Table 4.2: NOLH design with our factors and design points

BUDGET/ ALL.2019	ICOST/ E09427M.2018	WMR/ E09427M.ALL.2019	VALUE/ E09427M
33659146	633217.4	91	6.20839

30984247	509327.1	92	6.57121
31652972	540299.7	73	5.96650
32321696	571272.2	79	7.41781
38340219	622893.2	82	5.72462
41015118	519651.3	80	7.05498
37002770	499002.9	95	6.08744
36334045	612569.0	89	7.29687
35665320	550623.9	83	6.45027
37671494	468030.3	76	6.69216
40346393	591920.6	74	6.32933
39677669	560948.0	94	6.93404
39008944	529975.5	88	5.48273
32990421	478354.5	85	7.17593
30315522	581596.4	86	5.84556
34327871	602244.8	71	6.81310
34996595	488678.7	77	5.60367

3. Run ROME over the set of design points with ROMEfarmer

Now that we have all the files required by ROMEfarmer, i.e., ROMEmodel.zip and our DOE file, we can run our designed experiment.

The first thing to do is to create a study folder. We conveniently provide a “studies” folder in which you can create your individual study folders, but you should be able to create them in other locations. We will use the already created “sample_study” in the studies folder. You should copy your DOE .csv file into this study folder for ease of reference; again, it’s not required for ROMEfarmer, but an easy way to keep all the housekeeping straight, i.e., all the files for a particular experiment in one location. The sample_study includes the sampledoe.csv file we created above.

Although ROMEfarmer comes with a command-line interface (see source/ROMEfarmer.py for the arguments and usage), we will use the bare-bones Excel front-end. The front-end basically asks the user for the required information, constructs the appropriate command line, and calls ROMEfarmer. Figure 4.4 shows the interface when you open up ROMEfarmer-frontend.xlsm.

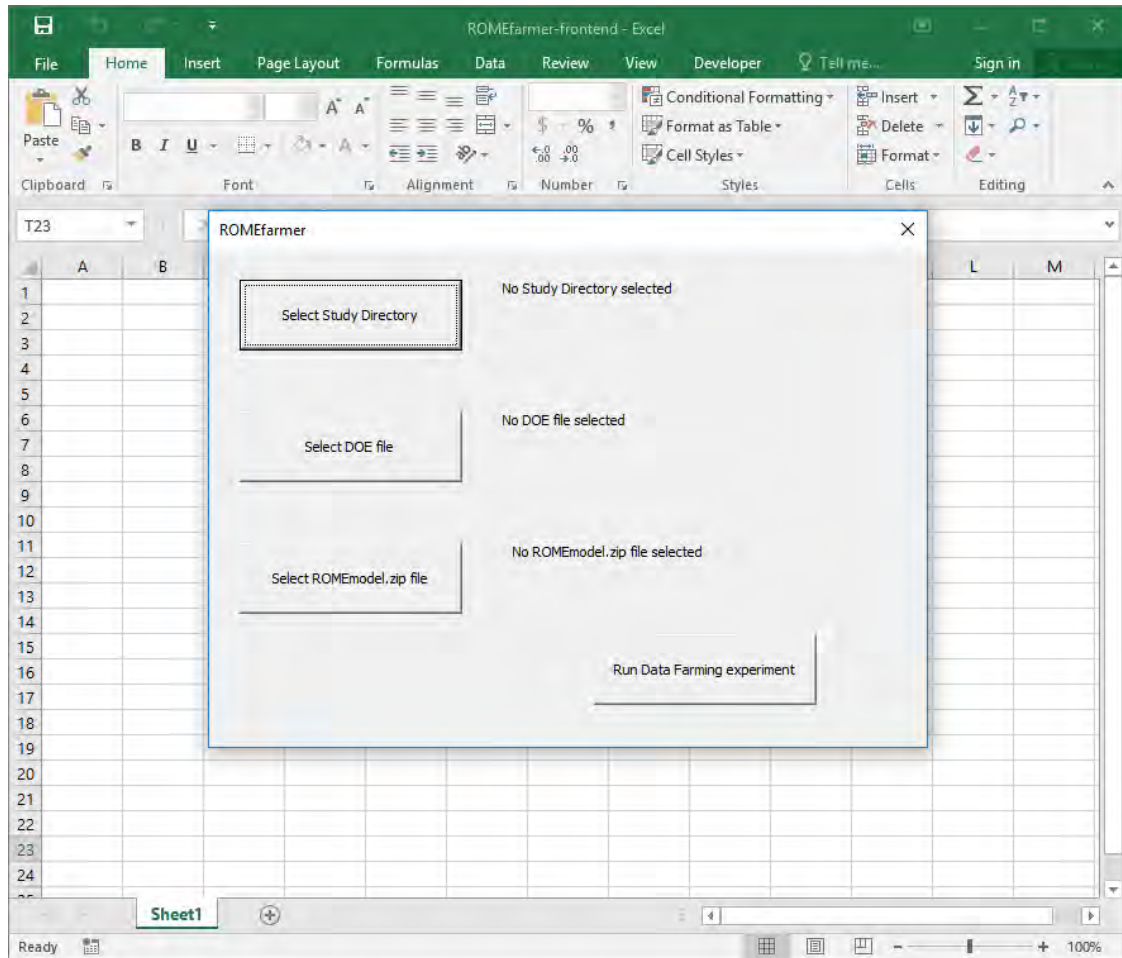


Figure 4.4: ROMEfarmer front-end

The user selects each button in turn (actually, the order isn't important among the three "Select.." buttons), filling out the necessary information, replacing the "No ..." with a specific folder or file. If the user fails to fill out any of the fields, ROMEfarmer won't start, i.e., the "Run Data Farming experiment" button will display an appropriate message indicating the missing information.

Once the user fills out all the information, the interface should look like Figure 4.5.

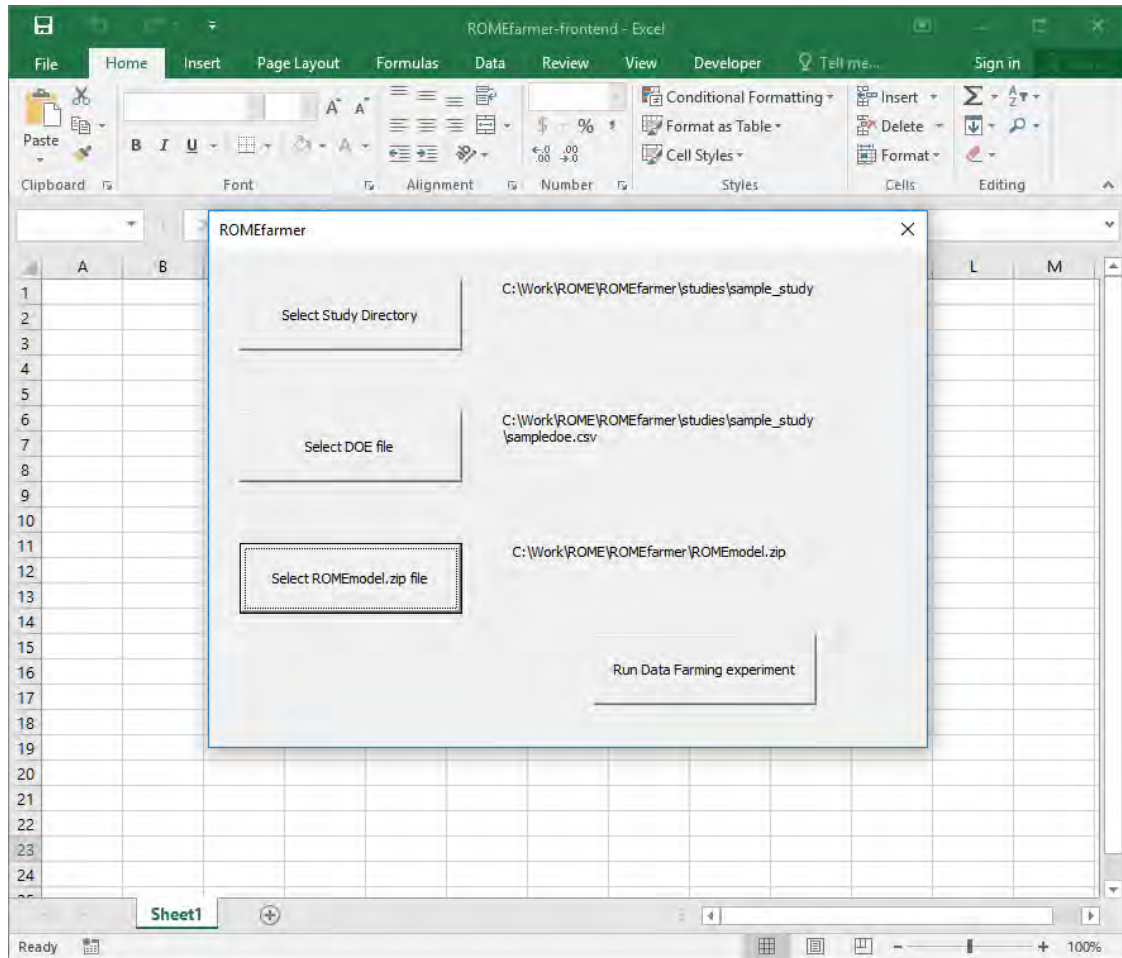


Figure 4.5: ROMEfarmer front-end-2

Once the three entries are completed, the user selects the “Run Data Farming experiment” button. The user is then shown a sequence of dialog boxes displaying the information that was given, and then ROMEfarmer is called. At that point, a Windows DOS window pops up, indicating the start, and progress, of ROMEfarmer. If all goes well, at completion, the DOS window pauses; this allows the user to view the final log entries of the experiment. The user then presses any key to continue, resulting in a final dialog box popping up indicating that the data farming experiment is complete and a message indicating where the output is stored. See Figure 4.6 for a screen shot of the DOS window.

```
C:\Windows\system32\cmd.exe

Done!
ROMEranner : DP-10::copying output in rundir: C:\Users\STEPHE~1\AppData\Local\Temp\tmpk3o97pfn to studydir output: C:\W
ork\ROME\ROMEfanner\studies\sample_study\Output
root       : DP-12::took 4.763746738433838 seconds to complete
root       : DP-17::starting run at Thu Aug  9 09:47:08 2018
ROMEranner : DP-17::running ROMEranner in study_dir:C:\Work\ROME\ROMEfanner\studies\sample_study
ROMEranner : created temporary directory: C:\Users\STEPHE~1\AppData\Local\Temp\tmpmltvczxx
ROMEranner : DP-17::copying file: C:\Work\ROME\ROMEfanner\studies\sample_study\sampladoe.csv to run dir: C:\Users\STEPH
E~1\AppData\Local\Temp\tmpmltvczxx
ROMEranner : DP-17::copying file: C:\Work\ROME\ROMEfanner\ROMEmodel.zip to run dir: C:\Users\STEPHE~1\AppData\Local\Temp
p\tmpmltvczxx
root       : DP-10::took 5.6671061515808105 seconds to complete
ROMEranner : DP-17::modifying input
InputModifier: DP-17::reading DOE file: C:\Work\ROME\ROMEfanner\studies\sample_study\sampladoe.csv
InputModifier: DP-17::changing factor: BUDGET/ALL.2019 to 34996595
InputModifier: DP-17::replacing line containing BUDGET/ALL.2019
InputModifier: DP-17::changing factor: ICOST/E09427M.2018 to 488678.7
InputModifier: DP-17::replacing line containing ICOST/E09427M.2018
InputModifier: DP-17::changing factor: WMR/E09427M.ALL.2019 to 77
InputModifier: DP-17::replacing line containing WMR/E09427M.ALL.2019
InputModifier: DP-17::changing factor: VALUE/E09427M to 5.60367
InputModifier: DP-17::replacing line containing VALUE/E09427M
ROMEranner : DP-17::running GAMS in C:\Users\STEPHE~1\AppData\Local\Temp\tmpmltvczxx

Done!
ROMEranner : DP-16::copying output in rundir: C:\Users\STEPHE~1\AppData\Local\Temp\tmpd1tdx1rk to studydir output: C:\W
ork\ROME\ROMEfanner\studies\sample_study\Output
root       : DP-16::took 5.282758474349976 seconds to complete
```

Figure 4.6: ROMEfanner front-end-3

All the information that was displayed in the DOS window is captured in a log file that is saved in the study folder. Additionally, in the study folder, an Output folder is created that has all the GAMS and ROME output, as well as a GAMS log file. Each of the output filenames are prefaced with a “DP-” and a number indicating the row or design point associated with that output file.

ROMEfanner works by running several design points concurrently, depending on the number of processors on your computer, and runs until all design points have been processed. It starts by creating a log file for the run and creates the Output folder in the study folder. Then for each design point, it calls ROMEranner, which runs a single design point.

ROMEranner then:

1. creates a temporary work directory
 2. copies the ROMEmodel.zip and the DOE file to the work directory
 3. unzips the ROMEmodel.zip in the work directory
 4. reads the DOE file and modifies the ROME input files based on the design point
 5. runs the ROME model
 6. copies the deroOut.csv, dero.lst, and dero.log file to the study Output folder, renaming them with the design point number
 7. completes by removing the temporary work directory
4. Post-process and analyze the output

At the completion of all the runs, ROMEfarmer then calls ROMEminer to create two output files: one is a concatenation of all the output from the single DP runs into a single file; and, the other is a file containing the objective value with any errors encountered by GAMS, and merged with the DOE file. Both of these are comma-separated values (CSV) files that can then be imported into most analysis tools.

5. Repeat as desired

Typically, an analyst does not just run one designed experiment. Analysis and information from a designed experiment usually leads to additional questions, that result in follow-on designed experiments. An analyst then repeats the above process with a new DOE file, and possibly even changing the base case by incorporating a new set of input files.

6. 5 References

- ROME Accreditation Report
- SEED Center for Data Farming web page, <https://my.nps.edu/web/seed/> (for papers, software downloads, etc.)

7. 6 Appendix

8. 6.1 Installing python

There are a number of options for installing python.

9. 6.1.1 Basic python installation

A basic python installation for Windows will install the python interpreter and python's basic packages. You can follow the instructions on python.org's website [<https://www.python.org/>] or follow the instructions at the realpython website [<https://realpython.com/installing-python/>]. Once you've installed the python, you will also need to install the numpy and pandas packages, as well as their dependencies. For that, you can use the 'pip' module. See the documentation for further information.

10. 6.1.2 Anaconda distribution

Anaconda is a pre-packaged python installation that includes a number of python packages for data science, including numpy and pandas. To download and install, follow the instructions on Anaconda's website [<https://www.anaconda.com/download/>]. To

install other packages, Anaconda uses the ‘conda’ package manager. See the documentation for more details.

11.6.2 ROMEmodel.zip file structure

Below is a list of the required files needed in the ROMEmodel.zip file and with the needed structure. The first two files are at the top level of the zip file, the files listed under “input” need to be in a folder called “input,” and similarly for the “scripts” folder. The input files are typically generated by an outside process that pulls the data from various MARCORMARCORLOGCOM databases. The *.gms files basically make up the ROME optimization model, and the runDERO.bat file is a Windows DOS-batch file that calls GAMS. It includes a path to the GAMS installation on the user’s computer (currently set at “c:\GAMS\win64\23.8.” NOTE: If the user updates GAMS or moves the location of GAMS, this file needs to be edited appropriately.

- dero.gms
- runDERO.bat
- input
 - BUDGET.txt
 - DSPARE0.txt
 - FORCESET.txt
 - ICOST.txt
 - INTCPT.txt
 - ISSUE.txt
 - RPCT.txt
 - SBL.txt
 - SEGMENTSET.txt
 - SLOPE.txt
 - SRFI.txt
 - TAMCNSET.txt
 - UNFF0.txt
 - USPARE0.txt
 - USR.txt
 - VALUE.txt
 - WBS.csv
 - WMR.txt
 - WO.txt
 - YEARSET.txt

- scripts
 - calcddata.gms
 - makeOutput.gms

LIST OF REFERENCES

- Barger JM (1995) Optimal depot level maintenance planning. Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.
- Brown GG, Dell RF, Wood RK (1997) Optimization and Persistence. *Interfaces* 27(5):15–37.
- Coleman A (2018) Marine Corps Logistics Command Enterprise Overview. Presentation, January 20, Naval Postgraduate School, Monterey, CA.
- DeMarco MA (2014) *Repair Optimization Materiel Evaluator (ROME) User's Manual*. User's Manual, Marine Corps Logistics Command, Albany, GA.
- Department of Defense (2018) Operation and Maintenance Overview Fiscal Year 2019 Budget Estimates. Department of Defense, Washington, D.C.
- Department of the Navy (2016) Maintenance Operations, Marine Corps Technical Publication 3-40E, Washington, D.C., <http://www.marines.mil/Portals/59/Publications/MCTP%203-40E.pdf?ver=2017-03-27-080524-963>.
- Department of the Navy (2018) Marine Corps Readiness Reportable Ground Equipment. Marine Corps Bulletin 3000. Washington, D.C. <http://www.marines.mil/Portals/59/Publications/FY19%20MCBUL%203000%20V3.pdf?ver=2018-11-14-123017-337>.
- Drexler JA (2003) Enhancing persistence when optimally scheduling depot-level repair activity for the United States Marine Corps. Thesis, Department of Operations Research, Naval Postgraduate School, Monterey, CA.
- GAMS (2019) GAMS Standard Price List. GAMS Official website. Accessed March 23, 2019, https://www.gams.com/fileadmin/standard_price_list.pdf.
- Goodhart CA (1999) Depot-level maintenance planning for Marine Corps ground equipment. *Mil. Oper. Res.* 4(3):77–89.
- Headquarters United States Marine Corps (2016) Marine Corps Operating Concept, Washington, D.C., <https://www.mccdc.marines.mil/Portals/172/Docs/MCCDC/young/MCCDC-YH/document/final/Marine%20Corps%20Operating%20Concept%20Sept%202016.pdf?ver=2016-09-28-083439-483>.
- Hernandez A, Lucas TW, Carlyle M (2012) Constructing nearly orthogonal Latin hypercubes for any nonsaturated run-variable combination. *ACM Transactions on Modeling and Computer Simulation* 22:20:1-20:17.

- Lucas TW, Kelton WD, Sánchez PJ, Sanchez SM, Anderson BL (2015) Changing the paradigm: Simulation, now a method of first resort: Simulation, Now a Method of First Resort. *Nav. Res. Logist. NRL* 62(4):293–303.
- Mullins M, Adams T, Simms R (2005) Analysis of Light Armored Vehicle Depot Level Maintenance. MBA Professional Report, Graduate School of Business and Public Policy, Naval Postgraduate School, Monterey, CA.
- United States Code 10 § 2460 (2019). Definition of depot-level maintenance and repair.
- Upton SC (2018) *ROMEfarmer User's Manual*, version 0.3. User's manual, SEED Center for Data Farming, Monterey, CA.
- Vieira H, Sanchez SM, Kienitz KH, Belderrain MCN (2013) Efficient, nearly orthogonal-and-balanced, mixed designs: an effective way to conduct trade-off analyses via simulation. *J. Simul.* 7(4):264–275.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California