



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**USING DATA FARMING AND OPTIMIZATION
TO ENABLE ANALYSIS OF CONCEPTS OF
EMPLOYMENT FOR SURFACE CONNECTORS**

by

Forest Q. Sentinella

June 2021

Thesis Advisor:
Second Reader:

Thomas W. Lucas
Emily M. Craparo

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 2021	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE USING DATA FARMING AND OPTIMIZATION TO ENABLE ANALYSIS OF CONCEPTS OF EMPLOYMENT FOR SURFACE CONNECTORS			5. FUNDING NUMBERS W2122	
6. AUTHOR(S) Forest Q. Sentinella				
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 Combat Development Command (MCCDC), , Quantico VA 22134			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 requires models and tools to examine what mix of connectors is needed to deliver equipment, personnel, and supplies during or before the contact phase of a conflict. If hostilities ensue, the Marine Corps needs the capability to determine the optimal connector fleet and delivery schedule for those connectors to insert and sustain the distributed force. In 2019, the Operations Analysis Directorate, Combat Development & Integration, developed a Schedule Mixed Integer Program (S-MIP) that optimizes the delivery schedule for transporting a set of equipment from designated source points to demand points, given inputs such as connector quantities, speed, range, carrying capacity, ability to embark/debark at given ports (nodes), and sea state. As part of a Naval Research Program effort, NPS reviewed the original formulation and code and implemented improvements. This thesis supplements that effort by data farming the improved model to assess the impact of changes as well as to explore two solution strategies: single iteration and rolling horizon. We demonstrate that the improved model yielded a solution that delivered all serials in less time due to the more efficient use of decision variables, allowing for a greater number of previous decisions to be revisited using a rolling horizon. Further, the single iteration solve produced the solution with the lowest delivery time, and we therefore recommend continuing to experiment with this method.				
14. SUBJECT TERMS high-dimensional experimental design, first island chain, logistics, mixed integer program, optimized logistical model			15. NUMBER OF PAGES 85	
			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.

**USING DATA FARMING AND OPTIMIZATION TO ENABLE ANALYSIS
OF CONCEPTS OF EMPLOYMENT FOR SURFACE CONNECTORS**

Forest Q. Sentinella
Captain, United States Army Reserve
BS, Texas A & M University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF OPERATIONAL ANALYSIS
(WARFARE)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2021**

Approved by: Thomas W. Lucas
Advisor

Emily M. Craparo
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Marine Corps requires models and tools to examine what mix of connectors is needed to deliver equipment, personnel, and supplies during or before the contact phase of a conflict. If hostilities ensue, the Marine Corps needs the capability to determine the optimal connector fleet and delivery schedule for those connectors to insert and sustain the distributed force.

In 2019, the Operations Analysis Directorate, Combat Development & Integration, developed a Schedule Mixed Integer Program (S-MIP) that optimizes the delivery schedule for transporting a set of equipment from designated source points to demand points, given inputs such as connector quantities, speed, range, carrying capacity, ability to embark/debark at given ports (nodes), and sea state. As part of a Naval Research Program effort, NPS reviewed the original formulation and code and implemented improvements. This thesis supplements that effort by data farming the improved model to assess the impact of changes as well as to explore two solution strategies: single iteration and rolling horizon. We demonstrate that the improved model yielded a solution that delivered all serials in less time due to the more efficient use of decision variables, allowing for a greater number of previous decisions to be revisited using a rolling horizon. Further, the single iteration solve produced the solution with the lowest delivery time, and we therefore recommend continuing to experiment with this method.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	RESEARCH PROBLEM.....	1
B.	RESEARCH QUESTIONS.....	2
C.	BACKGROUND	3
D.	ORGANIZATION OF THE THESIS.....	10
II.	LITERATURE REVIEW	11
A.	LOGISTICS OPTIMIZATION MODELS	11
B.	DESIGN OF EXPERIMENTS	12
C.	DESIGN OF EXPERIMENTS APPLIED TO OPTIMIZATION MODELS	12
III.	MODEL FORMULATION AND IMPROVEMENTS	15
A.	SCHEDULE MIXED INTEGER PROGRAM FORMULATION.....	15
1.	Sets and Indices.....	15
2.	Data [units]	16
3.	Decision Variables [units].....	16
4.	Formulation	17
B.	SCHEDULE MIXED INTEGER PROGRAM INPUT DATA	19
1.	WhatWhere	20
2.	HowFar	20
3.	ConnectorData	21
4.	ExperimentDesign.....	22
C.	SINGLE SOLVE VERSUS ROLLING HORIZON SOLVE	22
D.	NPS IMPROVEMENTS TO THE SCHEDULE MIXED INTEGER PROGRAM.....	25
IV.	EXPERIMENTATION	27
A.	SOFTWARE.....	27
B.	DESIGN OF EXPERIMENTS	27
C.	EXPERIMENT 1	30
D.	EXPERIMENT 2	30
E.	HIGH PERFORMANCE COMPUTING.....	31
F.	ANALYTICAL SOFTWARE.....	32

V.	RESULTS	33
A.	COMPARISON OF RESULTS (REVISED CODE VERSUS ORIGINAL).....	33
B.	EXPERIMENT 1 (ROLLING HORIZON SOLVE).....	35
C.	EXPERIMENT 2 (SINGLE SOLVE).....	39
VI.	CONCLUSIONS AND FUTURE WORK.....	45
A.	CONCLUSION	45
B.	RECOMMENDATIONS.....	46
C.	FOLLOW-ON RESEARCH.....	47
	APPENDIX: CONNECTORFARMER USER’S MANUAL (V0.14) (UPTON 2021)	49
	LIST OF REFERENCES.....	59
	INITIAL DISTRIBUTION LIST	63

LIST OF FIGURES

Figure 1.	South China Sea Economic Exclusion Zones. Illustrates stakeholder interests within the FIC. Source: DeMarco (2021).	5
Figure 2.	This photograph depicts the struggles faced in 1950 and relates to present-day vulnerabilities. Source: Naval History and Heritage Command (1950).	7
Figure 3.	Seabasing Logistics Architecture Overview. Source: DON (2006).	9
Figure 4.	Snapshot of the WhatWhere worksheet, best viewed in color. Adapted from Freeman (2019).	20
Figure 5.	Snapshot of the HowFar worksheet, best viewed in color. Adapted from Freeman (2019).	21
Figure 6.	Snapshot of the ConnectorData worksheet, best viewed in color. Adapted from Freeman (2019).	22
Figure 7.	Snapshot of the ExperimentDesign worksheet, best viewed in color. Adapted from Freeman (2019).	22
Figure 8.	Rolling Horizons vs. Single Solve solution strategies, best viewed in color	24
Figure 9.	Capture the flag illustrating a sparse full factorial design (2^2) and a denser full factorial design (11^2). Source: Sanchez et al. (2020).	29
Figure 10.	Snapshot of ConnectorMiner output	34
Figure 11.	Histograms and summary statistics for individual serial delivery times for five runs of the original model (bottom) and one run of the revised model (top).	35
Figure 12.	Snapshot of ConnectorMiner output for Experiment 1	36
Figure 13.	Partition tree for time to deliver	36
Figure 14.	Partition tree for solver status	38
Figure 15.	Snapshot of ConnectorMiner output for Experiment 2	39
Figure 16.	Actual by predicted plot for proportion of serials delivered	40
Figure 17.	Regression plot for proportion of serials delivered	41

Figure 18.	Sorted parameter estimates and prediction profiler generated by JMP	42
Figure 19.	Proportion of serials delivered and time to deliver by VARLIM and mipgap.....	42
Figure 20.	Comparison of individual serial delivery time statistics (revised model versus original model).....	46
Figure 21.	Comparison of best solution and run time for the rolling horizon and single iteration runs.....	46

LIST OF TABLES

Table 1.	Summary of input data specified via the Excel workbook. Variables in red text indicate minimal requirements for the model to run	19
Table 2.	Descriptions of factors and levels for Experiment 1 involving rolling horizons	30
Table 3.	Descriptions of factors and levels for Experiment 2 involving single solve	31

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAM	Asset Allocation Optimization Model
CD&I	Combat Development & Integration
C-OPT	Connector Optimization
DOD	U.S. Department of Defense
DOE	Design of Experiment
EABO	Expeditionary Advanced Based Operations
EEZ	Exclusive Economic Zone
FIC	First Island Chain
HADR	Humanitarian Assistance and Disaster Relief
IP	integer program
JLOTS	Joint Logistics Over the Shore
JOA	Joint Operating Area
LOCE	littoral operations in contested environments
LOGCOM	Logistics Command
MACS	Marine Expeditionary Unit Amphibious Connector Scheduler
MCCDC	Marine Corps Combat Development Command
MCDP	Marine Corps Doctrinal Publication
MIP	mixed integer programming
NATO	North Atlantic Treaty Organization
NPS	Naval Postgraduate School
NRP	Naval Research Program
OAD	Operations Analysis Directorate
POD	port of debarkation
POE	port of embarkation
PRC	People's Republic of China
ROME	Repair Optimization Materiel Evaluator
SEED	The Simulation Experiments and Experimental Design
S-MIP	Schedule Mixed Integer Program
SPD	shortest-path distance
SPOD	Sea Port of Debarkation

SPOE	Sea Port of Embarkation
SSTP	Ship to Shore Transportation Problem
TLAM	Tomahawk land attack missile
TTP	Tactics, techniques, and procedures
UAV	unmanned aerial vehicle
UNCLOS	United Nations Convention for the Law of the Sea
USMC	United States Marine Corps

EXECUTIVE SUMMARY

As state actors continue to develop sea denial capabilities and capacities, the potential for confrontation is growing in littoral environments. A common technique used by the United States is forward positioned forces as a method to enforce international norms. The Marine Corps is exploring the use of Expeditionary Advanced Based Operations (EABO) as a means of deterring aggression and reassuring allies. Sustainment of the force is critical in determining the depth of decisive operations that can be conducted and to enhance the ability to seize, retain, and exploit the initiative. An important question is—what is the logistical burden of EABO during times of conflict? The Marine Corps needs models and tools to evaluate delivery concepts for insertion and sustainment of personnel, equipment, and supplies.

In 2019, Major Nicholas Freeman from the Operations Analysis Directorate, Combat Development & Integration, developed the Schedule Mixed Integer Program (S-MIP), which optimizes the delivery schedule for transporting a set of equipment from designated source points to demand points, given inputs such as connector quantities, speed, range, carrying capacity, ability to embark/debark at given ports (nodes), and sea state. As part of a Naval Research Program effort, Naval Postgraduate School reviewed the original model formulation and code and implemented improvements.

This thesis investigated the original and improved models, conducted model runs via data farming, and produced insights into the following research questions:

- Do changes made to the original S-MIP model improve solution quality and run time?
- How might changes in the optimization solution strategy impact both solution quality and run time?
- Can data farming S-MIP provide us with a better understanding of the tool and increased confidence in its solutions?

Thesis work began with an examination and explanation of changes made to the S-MIP. Professor Emily Craparo of the Operations Research Department made several changes, described in more detail in Chapter III, including improving the rolling horizon implementation within the S-MIP code. An additional important improvement made by Mr. Stephen Upton of the Simulation Experiments & Efficient Designs (SEED) Center for Data Farming, NPS eliminated significant non-determinism in results previously exhibited by the model.

An experiment was conducted to compare the original model output to the improved model output, given a fixed set of inputs. The result of the experiment showed that the improvements resulted in an improved (lower) transportation time, with the additional benefit of eliminating the need to run replications of the model. A summary is provided in Figure 1.

Best Solution Found with Revised Model Versus Original Model					
Model	VARLIM	OVERLAP	All Serials Delivered	Max Serial Delivery Time (days)	Median Serial Delivery Time (days)
Revised	50000	2	YES	19.4	9.9
Original	50000	1	YES	25.3	11.3

Figure 1. Comparison of runs with original code versus improved code

Next, because the equipment set input into the S-MIP is so large, this thesis also evaluated the effectiveness of two different optimization solution strategies, one using a single iteration solve and another using a multi-iteration rolling horizon solve. These two strategies provide different advantages, such as reduced run time or improved optimality. To our knowledge, the S-MIP had not yet been evaluated using the single iteration solve strategy; therefore, the experimentation performed in this thesis provides valuable insight on useful settings, such as how large VARLIM needs to be and how much time it takes to run. A visual illustration of the use of a rolling horizon solve versus a single iteration solve is given in Figure 2, described in detail in Chapter III.

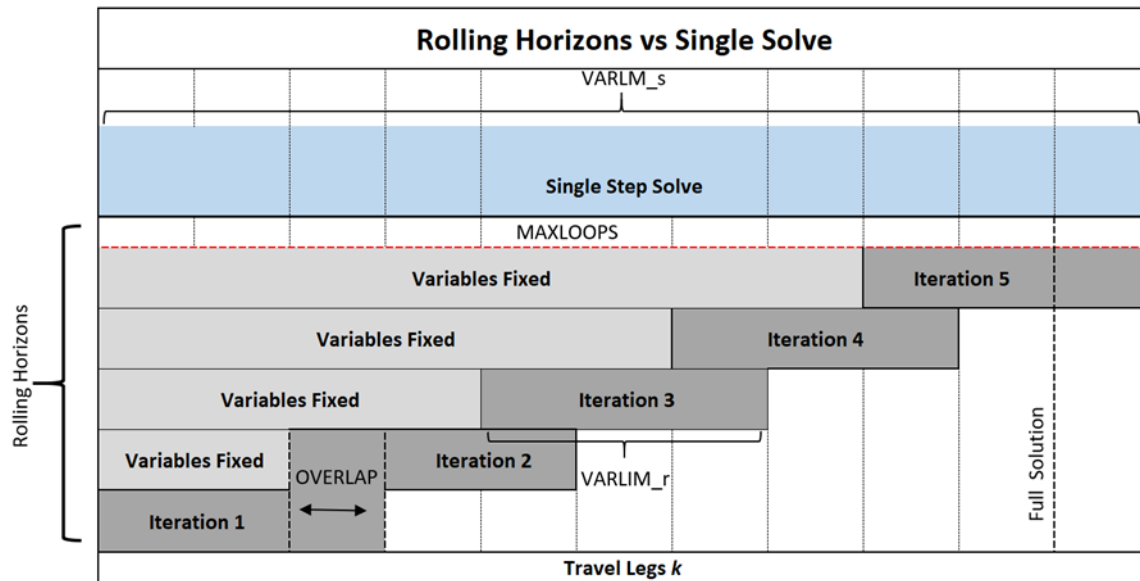


Figure 2. Rolling Horizons vs. Single Solve solution strategies, best viewed in color

Briefly, when solving with a single iteration, the number of variables required, and thus computational time, can be so large as to be prohibitive. One strategy to address this issue is to break the problem into several smaller problems, and to solve each one in succession, and that is what is implemented with a rolling horizon solve. Each rolling horizon iteration represents a single step in this multi-step solution process.

Each solution strategy was explored via an experiment that varied key parameters. Of the runs conducted for this study, the single iteration runs produced the best solution, delivering all serials of equipment in 21.6 days, but did require approximately five times longer to run than the best rolling horizon run. This result is summarized with Figure 3.

Multi Iteration Rolling Horizon Versus Single Iteration Solve		
Stratgey	Best time to deliver (days)	Run time (hours)
Rolling Horizons	23	44.97
Single Solve	21.6	232.56

Figure 3. Comparison of best solution and run time for the rolling horizon and single iteration runs

In summary, data farming the S-MIP provided a better understanding of the model and a better understanding of parameters that yield good results with each solution strategy employed. Based on the results of the runs conducted, we recommend continuing to experiment with use of the single iteration solve. So far, it has produced the best solution (delivering all serials in 21.6 days) among the runs conducted for this study. To potentially improve the solution even further while possibly reducing run time, we specifically recommend experimenting with the use of multiple CPLEX threads and running in opportunistic mode (parallel CPLEX parameter equal to either 0 or -1). Because the best solution strategy and CPLEX parameters are instance-specific, major changes to equipment set, connectors, or other inputs, would warrant performing additional experiments before transferring to a planner for use.

We conclude with a set of specific recommendations for further study, including: (1) developing a heuristic to provide a feasible solution with which to “warm start” the model; (2) running larger experiments that vary connector mixes and other operational inputs, utilizing efficient designs of experiment; and (3) exploring utilizing both simulation and optimization in a complementary way to improve decision support.

ACKNOWLEDGMENTS

I come from an extended family of military professionals, and I would like to thank them one and all for their influence, wise counsel and insight, which can only be gained by experience. Specifically, my father Col Derek Sentinella, a combat engineer; my brother Major Derek Sentinella, also a combat engineer who has 11 years' service with 58 months spent in the Middle East; my mother, Susan Sentinella, a special type of woman who has endured deploying both her husband and her children; Col John Hutcheson, who served on the Berlin Wall, for his constant mentorship and English lessons; Amanda Jones, Ryan Lin and Ian Henry, exceptional classmates whose families treated me like family (the tears and late-night meals); and finally, my dogs Sam and Daisy for insisting on study breaks and long walks on the beach. You helped keep me sane.

So many people made a contribution to the creation of this thesis. A special thanks to the NPS staff: Betsy Wallace for reviewing every single document I have submitted; Army Geospatial Center, Military Support Team 21 for creating the imagery of the FIC; Mary McDonald for reviewing the content of every single page; Dr. Pilnick for the opportunities that you have provided; and my thesis advisors Dr. Tom Lucas and Dr. Emily Craparo. Completing this thesis has truly been a journey. I could not have completed it without your guidance.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

In this chapter, we present the research problem and questions, pertinent background, and the organization of the thesis.

A. RESEARCH PROBLEM

As state actors continue to develop sea denial capabilities and capacities, the potential for confrontation has been growing in littoral environments. Some nations are attempting to establish control over neighboring coastal territories and disputed geography by an increasing pattern of aggressiveness (Concepts Branch, Marine Corps Warfighting Laboratory/Futures Directorate 2016, p. 6). The United States has long maintained forward-positioned forces afloat and ashore to reassure friends, deter aggression, respond to crises, maintain alliances, and enforce international norms (Department of the Navy [DON] 2021).

The Department of Defense (DOD) Dictionary of Military and Associated Terms defines the littoral environment as comprising of:

Two segments of operational environment: 1. Seaward: the area from the open ocean to the shore, which must be controlled to support operations ashore. 2. Landward: the area inland from the shore that can be supported and defended directly from the sea. (U.S. Department of Defense [DOD] 2021)

Both must be controlled to support operations ashore. This maritime terrain containing choke points as well as narrow and shallow seas is being combined with access to modern sensors and precision missiles with ranges that extend both seaward and landward by hundreds of miles. To address the new contested littoral environment, the Navy-Marine Corps recognizes and is developing strategies to support joint forces in littoral operations in contested environments (LOCE).

To support LOCE, Marine Corps Combat Development Command (MCCDC) is developing the concept of expeditionary advanced base operations (EABO). EABO places Marine Corps units forward within contested littoral environments. EABO is driven by an adversary's ability to deploy long-range precision fires in support of sea denial or counter-

intervention against U.S. and Coalition forces. EABO provides a mobile, low-cost, land-based capability necessary to respond within the contested littoral operational environment (DON 2019).

B. RESEARCH QUESTIONS

Marine Corps Doctrinal Publication (MCDP) 4, Logistics, states: “logistics provides the resources of combat power, positions those resources on the battlefield, and sustains them throughout the execution of operations (DON 2018).” The Commandant’s Planning Guidance poses the question: “[a]re EABO contributions to the joint force worth the logistical burden?”

Sustainment of the force is critical in determining the depth of decisive operations that can be conducted and to enhance the ability to seize, retain, and exploit the initiative. The Marine Corps needs models and tools to examine what mix of connectors (transport platforms) are needed to deliver equipment, personnel, and supplies during the contact phase of a conflict. In addition, the Marine Corps needs the ability to determine a connector fleet that is capable of sustaining a distributed force if hostilities ensue with peer or near peer adversaries. One such tool that the MCCDC is developing and using for these purposes is the Schedule Mixed Integer Program (S-MIP) (Freeman 2019).

This thesis uses optimization and design of experiments to gain insights into the following research questions:

- Do changes made to the original S-MIP model improve solution quality and run time?
- How might changes in the optimization solution strategy impact both solution quality and run time?
- Can data farming S-MIP provide us with a better understanding of the tool and increased confidence in its solutions?

This research utilizes experimental design techniques to enable exploration of S-MIP’s solution space as a function of model formulation choices and input parameters such

as connector capabilities and sea states. This includes the capability to vary operational parameters within S-MIP in a future tactical situation and to explore the ability of various mixes of surface ships to deliver equipment, personnel, and supplies during the contact phase of a Marine Corps operation in the First Island Chain (FIC) in the South China Sea. The research demonstrates the use of S-MIP to support analysis of a sponsor-provided scenario, and documents challenges and opportunities in using S-MIP in support of concept development activities.

The improved optimization model will assist the Marine Corps in determining the optimal logistics deployment to sustain the distributed force in the event of hostilities. Additionally, this research may guide Navy and Marine Corps acquisition and force structure decisions and/or recommend options for future research and development.

C. BACKGROUND

The following paragraphs introduce the political environment within the FIC and provide a current political example, effectively illustrating the envisioned future conflict environment and the current and evolving state of U.S. doctrine. Additionally, Operation Chromite serves as a historical logistical vignette of the last time U.S. amphibious doctrine was tested by an oppositional force. This section also introduces the concept of “seabasing,” and other key terms used throughout this thesis.

Freedom of the seas is of paramount survival interest to the United States and to the larger global economy of the free world. Maritime trade continues to be the cost-effective method for global transportation of goods, with global sea trade beginning and ending within the littorals (Vego 2015, pp. 30–76).

The United Nations Convention for the Law of the Sea (UNCLOS), established in December of 1982, provides a 200-nautical-mile Exclusive Economic Zone (EEZ) to every nation with a shoreline. UNCLOS states that within the EEZ the coastal state has rights for “exploring and exploiting, conserving, and managing the natural resources, whether living or non-living, of the waters superjacent to the seabed and of the seabed and its subsoil.” Therefore, challenges to the maritime commons are closely monitored by state actors and

could give rise to belligerents in areas such as the Western Pacific, Baltics, and Eastern Mediterranean.

The FIC is a littoral region with multiple stakeholders, man-made islands, and extensive global trade. The FIC's numerous closely grouped islands are separated by narrow seas that form a natural barrier between the People's Republic of China (PRC) mainland and the Pacific Ocean. At the very southern tip is the stretch of the South China Sea between Vietnam and Malaysia. An important global strategic and economical feature at the southern reaches of the FIC is the Strait of Malacca. As a narrow and shallow feature, the Strait of Malacca is a choke point of global economic, political, and military importance (Vego 2015, pp. 30–76).

An example of LOCE being supported by EABO could occur within the FIC, an area of strategic importance for both the United States and the PRC. The PRC is challenging UNCLOS, or the concept of the seas as a maritime commons, by claiming that territorial waters extend to the nine-dash line. The nine-dash line is a cartographical feature from a 1947 map that encompasses a substantial portion of the South China Sea (see Figure 1). Many stakeholders are impacted by the concept of the nine-dash line, including: users of global assets such as the Strait of Malacca, individual nations utilizing the EEZs, and U.S. assets such as Guam and Hawaii. This is an example of a potential adversary that can operate close to home territory with an established multilayered defense with long-range sensors and precision missiles, thereby exploiting the maritime terrain to deny freedom of movement.

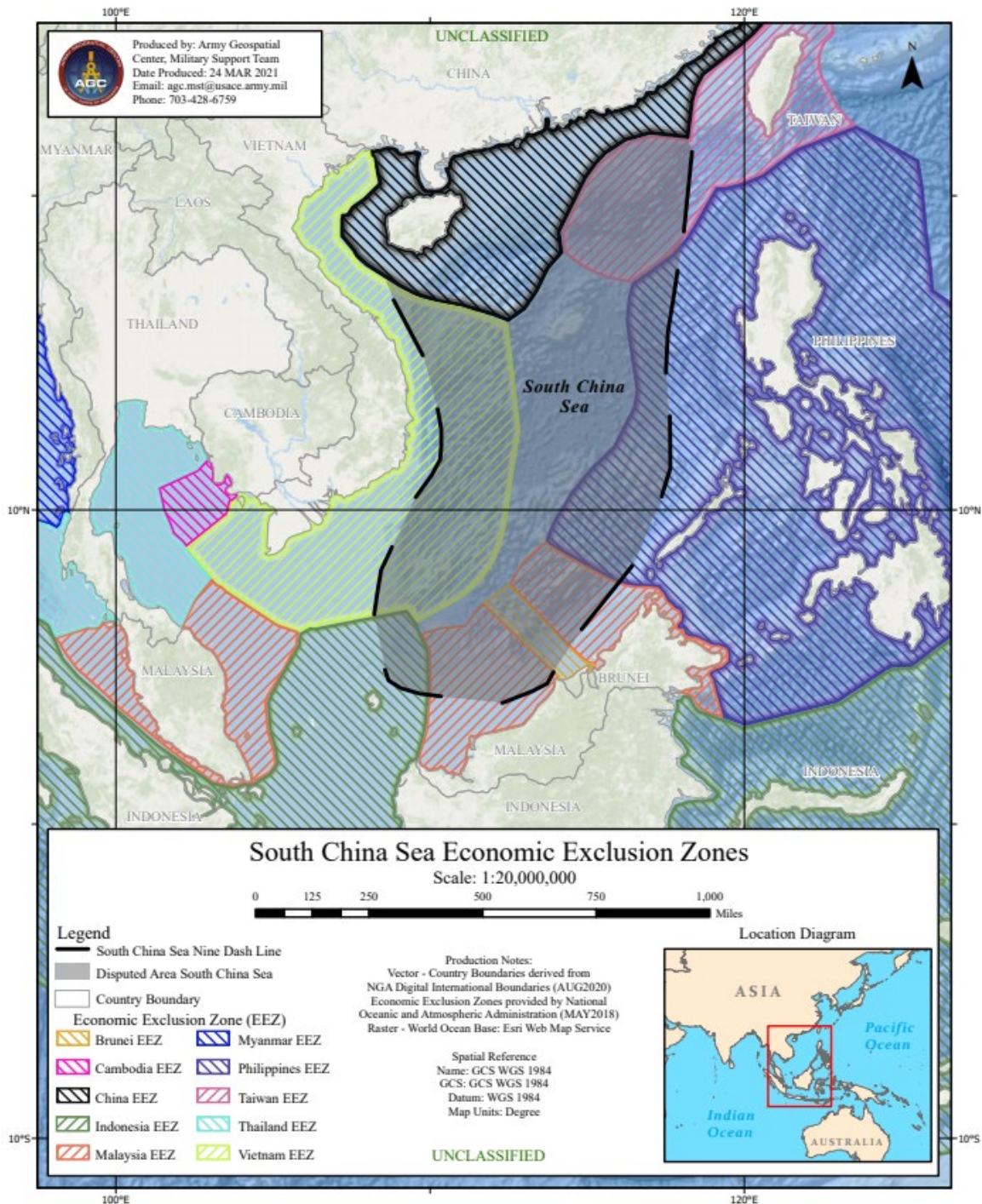


Figure 1. South China Sea Economic Exclusion Zones. Illustrates stakeholder interests within the FIC. Source: DeMarco (2021).

Before looking ahead to potential future LOCE scenarios, it is worthwhile to reflect on past operations. The largest opposed amphibious operation conducted by U.S. Forces since WWII was Operation Chromite (15 Sept 1950). Since Operation Chromite occurred at Incheon, South Korea, just five years post WWII, the institutional knowledge from amphibious landings executed during WWII was present in U.S. forces. The objective of Operation Chromite was to damage the North Korean People's Army supply network and cut off the supply lines to oppositional forces operating in present-day South Korea. This historical vignette is complete with a combination of forces that included: all branches of U.S. forces; multinational, UN, and local national forces; harbor limitations such as a large tidal range (+/- 30 feet), extreme tides, day-time landings, high sea walls, and mud flats; available hard structures in varying degree of service; and bad weather, including potential typhoons, element of surprise, redundancy of ships, supply shortages, and frustrated cargo. MAJ Todd Zwolensky (2007) effectively argues the importance of logistical support during Operation Chromite in his thesis titled: "Logistical Support for amphibious landings is critical to the success of this operation due to the inherent offensive nature and operational scope."



“Four LSTs unload men and equipment while high and dry at low tide on Inchon’s Red Beach, 16 September 1950, the day after the initial landings there. USS LST-715 is on the right end of this group, which also includes LST-611, LST-845 and one other. Another LST is beached on the tidal mud flats at the extreme right. Note bombardment damage to the building in center foreground, many trucks at work, Wolmi-Do island in the left background and the causeway connecting the island to Inchon. Ship in the far distance, just beyond the right end of Wolmi-Do, is USS Lyman K. Swenson (DD-729).” (Naval History and Heritage Command 1950)

Figure 2. This photograph depicts the struggles faced in 1950 and relates to present-day vulnerabilities. Source: Naval History and Heritage Command (1950).

To support Operation Chromite’s offensive effort, the capability now known as joint logistics over the shore (JLOTS) was utilized. Joint Publication 4-0 defines JLOTS as the “process of loading and unloading of ships without the benefit of deep draft-capable, fixed port facilities; or as a means of moving forces closer to tactical assembly areas” (DOD 2019). As present-day potential belligerents continue to explore stand-off engagement capabilities, an identified growing vulnerability is occurring in the replenishment process

and procedures that transport sustainment to joint ground forces. JLOTS is typically conducted in low-threat environments utilizing the Army to provide the supply structure for forward operating ground forces. The vulnerability is the requirement for hard structures such as airports, seaports, and warehouses to ensure the transportation and distribution of sustainment. Operation Chromite was the last time U.S. Forces doctrine was tested at scale. Its tactics, techniques, and procedures proved successful in September 1950, but in the current operational environment they need to adjust to the threat of stand-off engagement capabilities and emerging technologies.

Seabasing is a new concept and capability that can serve to close this gap. *Seabasing Joint Integrating Enabling Concepts* defines seabasing as the

rapid deployment, assembly, command, projection, reconstitution, and re-employment of joint combat power from the sea, while providing continuous support, sustainment, and force projection to expeditionary Joint forces without reliance on land bases within the Joint Operating Area (JOA). (DON 2006)

Seabasing is designed to expand operational maneuver and facilitate assured access and entry from the sea by creating an additional repositionable logistical node (DON 2006). This logistical node assists in minimizing equipment, supplies, and infrastructure ashore, thereby reducing the inland footprint and maximizing the use of international waters, while providing sustainable logistical functions to Joint forces both on shore or at sea. This research explores important seabasing concepts that directly impact timely and persistent resupply.

Figure 3 depicts both the architecture of seabasing logistics, and the terminology used throughout this research (DON 2006). “Advance base,” “seabase,” and “objective” are called “nodes.” Advance base is the primary warehouse that provides support to the seabase and is located outside of the JOA and less than 2000 nautical miles from the seabase. The seabase is a repositionable node that provides support and sustainment to expeditionary forces. “Connectors” are the logistical vessels that carry personnel, equipment, and sustainment (serials) between the nodes’ traveling routes called “arcs” (arrows) that connect the nodes. A collection of arcs, occurring between a port of embarkation (POE) and a port of debarkation (POD), are collectively referred to as a

“transit.” This research focuses on the mix of connectors that operate on the transit (green arrows) between the seabase node and the objective node.

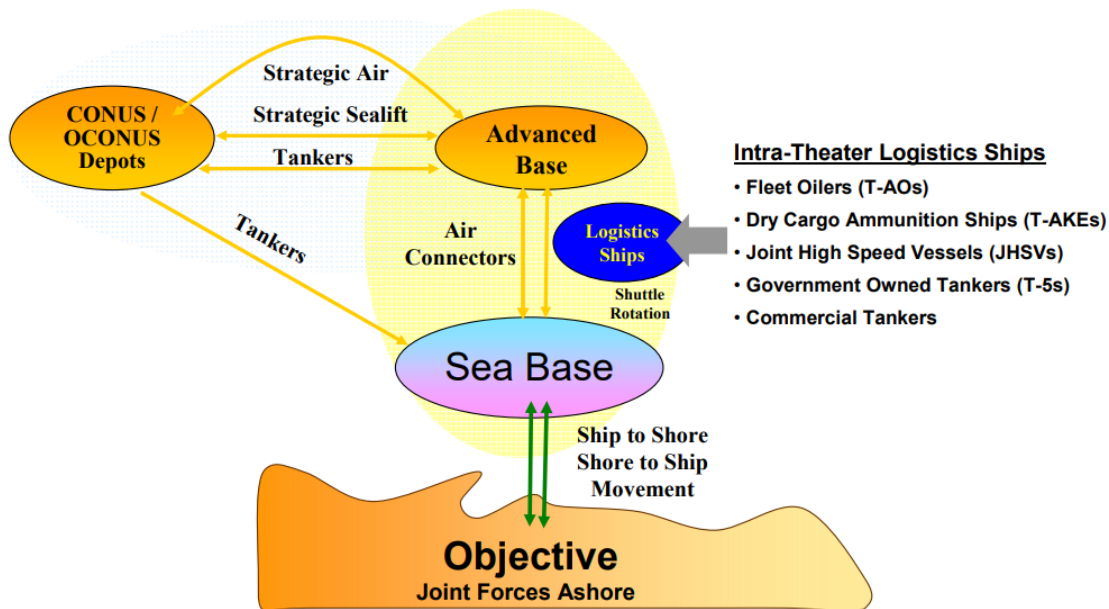


Figure 3. Seabasing Logistics Architecture Overview. Source: DON (2006).

To support the EABO concept, the Marine Corps requires logistical support to meet the consumption rates of multiple EABOs during the “seize the initiative” phase of operations. During the seize the initiative phase, substantial ammunition and other logistical resupply is needed to enable the force to hold, advance while in contact, or engage with opposition forces. A deciding factor in many campaigns has been logistics. Dwight D. Eisenhower stated, “You will not find it difficult to prove that battles, campaigns, and even wars have been won or lost primarily because of logistics.” (The Heritage Foundation 2019) A logistical network must operate effectively, remain robust to uncertainty, and provide the right quantity of materiel at the right time in an area of conflict. Expeditionary operations have the smallest room for error and thus it is strategically essential to implement a robust logistical network with an adequate mix of connectors.

D. ORGANIZATION OF THE THESIS

The remainder of this thesis is as follows: Chapter II, the literature review, discusses previous research related to logistics optimization models as well as design of experiments (DOE). Chapter III, methodology, discusses S-MIP model developed by Major Freeman at MCDDC, the changes made to S-MIP, and the experiments designed to measure the impact of these changes. In the analysis chapter, Chapter IV, the experiment results are analyzed using a variety of statistical and graphical methods. The conclusion, Chapter V, summarizes the main results of this research and recommends potential avenues for follow-on research.

II. LITERATURE REVIEW

In this chapter, we present a brief overview of selected, pertinent research related to logistics optimization models as well as design of experiments (DOE) and its application to optimization models.

A. LOGISTICS OPTIMIZATION MODELS

There are numerous examples of applying optimization to determine how best to deliver supplies from a source point to demand points. Three examples of research conducted at the Naval Postgraduate School (NPS), that use optimization to understand how best to deliver Navy or Marine Corps supplies from ship to shore, follow.

Peter Ward's 2008 thesis investigates optimizing ship-to-shore movement for hospital ships during humanitarian assistance and disaster relief (HADR) operations (Ward 2008). The Ship to Shore Transportation Problem (SSTP) refers to the requirement of transporting personnel and patients between hospital ships (T-AH) via connectors such as helicopters, watercraft, and ground vehicles during HADR. In Ward's research, SSTP is formulated into a mixed-integer mathematical optimization model. The SSTP objective function minimizes cost with decision variables of mission performance, personnel strength, and transportation asset utilization. Constraints account for unique aspects to T-AH such as aircraft limitations (flight deck) and watercraft (restricted embarkation and debarkation).

Angel Burgos and Gary D. Mclean Jr's 2018 thesis focuses on HADR operations occurring in the Pacific theater and analyzes planning factors critical to different ship classes (Burgos A & Mclean GD 2008). Burgos and Mclean utilize the Joint Transportation Optimization Planner – Sealift (JTOP-S) model to determine an optimal solution for variables such as port functionality, ship capacity, supply, and demand requirements unique to HADR operations. This thesis provides valuable insight regarding ship capacity and speed as limiting factors, and capacity of the Sea Port of Embarkation (SPOE) relative to the Sea Port of Debarkation (SPOD).

Finally, in 2017, Robert M. Christafore Jr. differs from the above theses and actively discusses challenges presented by seabasing to maritime planners (Christafore 2017). The author uses the Marine Expeditionary Unit Amphibious Connector Scheduler (MACS) to optimize and generate a feasible ship-to-shore schedule to deliver bulk fuel. Christafore’s model is designed to calculate the optimal number of round trips to meet the fuel demands of MEUs.

B. DESIGN OF EXPERIMENTS

DOE techniques enable the ability to exercise a model over numerous inputs, to gain insights into how the model is working, key drivers of performance, and threshold points—insights that would be difficult or impossible to obtain if we limited ourselves to a small number of ad hoc runs. The Simulation Experiments & Efficient Designs (SEED) (SEED) Center for Data Farming at NPS (<https://harvest.nps.edu>) specializes in providing a variety of efficient and flexible experimental designs to meet a variety of analytic goals for high-dimensional computational models.

The SEED Center uses the metaphor of “data farming” to describe iterative design and analysis of computer experiments (Lucas et al. 2015). Just like a farmer cultivates a plot of land to maximize yield, a data farmer intentionally and effectively manipulates simulation inputs using sound DOE techniques, to maximize information gained from experimentation. The data farmer thereby “grows” data needed for their analysis, according to their carefully designed experimental plan (Kleijnen et al. 2005). Kleijnen et al. provide a concise and informative overview of the benefits to be gained by data farming any model that takes inputs and produces outputs, whether that model be a simple spreadsheet model, combat simulation, or optimization model. Though the approach to designing the experiment may depend on the nature of the model, data farming vastly improves the information and insights possible from running any computational model.

C. DESIGN OF EXPERIMENTS APPLIED TO OPTIMIZATION MODELS

There are many potential benefits of applying DOE to optimization models, and these include the ability to:

- Explore the impact of uncertain parameters on the optimal objective value as well as on characteristics of the solution.
- Find solutions that are robust to these uncertainties.
- Analyze relationships between parameters, optimal values, and solution characteristics.
- Illuminate tradeoffs that might be involved.
- Investigate parameter settings (or combinations) that contribute to the chance of the solver finding a feasible solution, or the time required to find solutions.
- Systematically explore the existence and characteristics of multiple optimal solutions.
- Support verification and validation efforts.

We next describe two notable NPS theses that apply DOE to an optimization model. First, in 2015, LCDR Maxine J. Gardner's thesis utilized data farming to run a Navy HADR optimization model over a set of parameter inputs to develop insights into what constitutes a robust logistical response (Gardner 2015). Specifically, Gardner developed the Asset Allocation Optimization Model (AAM) and applied data farming to better understand how best to deploy disaster relief, given uncertainty about the nature of a particular type of natural disaster. This model provides valuable insight into the potential future usage of unmanned aerial vehicles (UAVs) in future HADR operations.

In 2019, Capt Thomas P. McKavitt's worked with the SEED Center for his thesis in which he ran a mixed integer programming (MIP) optimization model known as the Repair Optimization Materiel Evaluator (ROME) over a set of inputs of interest to the USMC Logistics Command (LOGCOM) (Mckavitt 2019). Specifically, McKavitt examined 15 different factors (parameters of the optimization model) impacting principal end items requiring depot level repair during a given fiscal year. McKavitt's research findings showed a bias occurring within the model, which affected the solution. McKavitt's

work serves as an example of the scrutiny that an optimization model requires and provided an opportunity to improve the model for future research.

III. MODEL FORMULATION AND IMPROVEMENTS

This chapter covers the formulation of S-MIP and the inputs it requires, as well as improvements that NPS made to the code. The model was implemented using the Python pyomo package. The user has a choice of solver, but we used the CPLEX Optimization Studio (IBM 2006).

A. SCHEDULE MIXED INTEGER PROGRAM FORMULATION

To help address the issue of logistics in support of EABO, Major Nicholas Freeman of the Current Operations Analysis Support Team (COAST), Operations Analysis Directorate (OAD), Combat Development & Integration (CD&I), developed a mixed integer program (MIP) called the Schedule Mixed Integer Program (S-MIP) in December of 2019. A MIP is a form of mathematical optimization in which some of the decision variables are continuous and others must be integer-valued. Major Freeman's S-MIP was designed to optimize the delivery of the serials (packages of equipment) from their source to their destination, given a specified travel network and set of connectors available to transport the serials (Freeman, 2019).

There is no previous published or openly available work that has used S-MIP. However, Dr. Kirk Yost of The MITRE Corporation expanded on the original formulation to account for the risk of detection along the arcs. The MITRE model formulation is called the Connector Optimization (C-OPT) model.

The following section provides the complete formulation of the original S-MIP (Freeman, 2019).

1. Sets and Indices

$v \in CXRS$	connectors
$s \in EQUIP$	serials
$i, j \in NODES$	nodes/locations
$(v, i, j) \in TRIPS$	arcs/transits executed by connectors

$k \in LEGS$	individual arcs or legs executed by the connectors (first leg, second leg, etc.)
$(v, k) \in CLEGS$	connector v executes at least k legs
$(v, k, i, j) \in S_TRIPS$	feasible connector transits
$(s, v, k, i, j) \in LOADS$	feasible connector-serial transits
$(s, i) \in PUTS$	nodes reachable by serial s

2. Data [units]

h_v	usable deck area for connector v [sq.ft]
v_s	value/priority weight for serial s
a_s	deck footprint size of serial s [sq.ft]
b_s	initial/starting location (node) for serial s
e_s	terminal/ending location (node) for serial s
$t_{v,i,j}$	time required for connector v to make transit (i, j) [days]
$LAYOVER$	time cost for connector to enter/leave nodes [days]
BIG	“infinite” = $ LEGS * (\max(t_{v,i,j}) + LAYOVER) * 10$
$STEP$	$=1, 2, \dots, MAXLOOPS$; the current successive time window

3. Decision Variables [units]

$W_{v,k}$	time ≥ 0 at which connector v completes leg k [days]
$X_{v,k,i,j}$	$=1$ if connector v makes transit (i, j) on leg k ; else 0
$Y_{s,v,k,i,j}$	$=1$ if serial s loaded for transit (v, k, i, j) ; else 0
$Z_{s,i}$	time ≥ 0 at which serial s arrives at node i [days]

P_s =1 if serial s not delivered to destination e_s ; else 0

U_s shortest-path distance ≥ 0 remaining for serial s to e_s [miles]

4. Formulation

$$\min_{W, X, Y, Z, P, U} \quad z = \sum_{s \in EQUIP} (STEP_v U_s + v_s Z_{s, e_s} + v_s P_s BIG) \quad (C1)$$

$$s.t. \quad \sum_{(v, k, j) : (s, v, k, i, j) \in LOADS} Y_{s, v, k, i, j} \leq 1 \quad \forall (s, i) \in PUTS \quad (C1)$$

$$t_{v, i, j} X_{v, k, i, j} + W_{v, k-1} + LAYOVER \leq W_{v, k} \quad \forall (v, k, i, j) \in S_TRIPS : k > 1 \quad (C2)$$

$$\sum_{(i, j) : (v, k, i, j) \in S_TRIPS} X_{v, k, i, j} \leq 1 \quad \forall (v, k) \in CLEGS \quad (C3)$$

$$\sum_{s : (s, v, k, i, j) \in LOADS} a_s Y_{s, v, k, i, j} \leq h_v X_{v, k, i, j} \quad \forall (v, k, i, j) \in S_TRIPS \quad (C4)$$

$$\sum_{j' : (v, j', i) \in TRIPS} X_{v, k-1, j', i} \geq X_{v, k, i, j} \quad \forall (v, k, i, j) \in S_TRIPS : k > 1 \quad (C5)$$

$$Y_{s, v, k, i, j} \leq \sum_{(v', k', j') : (s, v', k', j', i) \in LOADS} Y_{s, v', k', j', i} \quad \forall (s, v, k, i, j) \in LOADS : i \neq b_s \quad (C6)$$

$$W_{v, k} + (Y_{s, v, k, i, j} - 1) BIG + LAYOVER \leq Z_{s, j} \quad \forall (s, v, k, i, j) \in LOADS \quad (C7)$$

$$P_s \geq 1 - \sum_{(v, k, i, j) : (s, v, k, i, j) \in LOADS, j = e_s, (v, k, i, j) \in S_TRIPS} Y_{s, v, k, i, j} \quad \forall s \in EQUIP \quad (C8)$$

$$spd_{b_s} - \sum_{(v, k, i, j) \in S_TRIPS} Y_{s, v, k, i, j} (spd_i - spd_j) \leq U_s \quad \forall s \in EQUIP \quad (C9)$$

$$Z_{s, i} + t_{v, i, j} + (Y_{s, v, k, i, j} - 1) BIG + LAYOVER \leq W_{v, k} \quad \forall (s, v, k, i, j) \in LOADS \quad (C10)$$

$$W_{v, k}, Z_{s, i}, U_s \geq 0 \quad \forall v, k, s, i \quad (C11)$$

$$X_{v, k, i, j}, Y_{s, v, k, i, j}, P_s \in \{0, 1\} \quad \forall v, k, s, i, j \quad (C12)$$

Decision variables are given by $W_{v, k}$, $X_{v, k, i, j}$, $Y_{s, v, k, i, j}$, $Z_{s, i}$, P_s , and U_s . These include binary variables that determine whether or not a connector travels from node i to node j on leg k and whether or not a particular serial is transported by a given connector from node i to node j on leg k . Each connector will travel at most $|LEGS|$ legs, each time traveling from one node to another, either carrying serial(s) or traveling empty on its way to its next pickup point.

In more detail, $W_{v, k}$ is the time at which connector v completes leg k of the transit in days; $X_{v, k, i, j}$ is a binary variable that determines whether connector v will travel from i to

j on leg k ; $Y_{s,v,k,i,j}$ is a binary variable that determines whether serial s is loaded for transport by connector v from i to j on leg k ; $Z_{s,i}$ is the time at which serial s arrives at node i ; P_s is a binary variable indicating that serial s is not delivered during the planning horizon; and U_s is the shortest path distance in miles between the last location of serial s and its final destination.

The objective function minimizes a series of penalties, where the three penalty terms capture: (1) distance to destination for any equipment that doesn't arrive at its final destination, (2) the time to deliver equipment to its destination, and (3) a penalty for equipment that doesn't reach its final destination.

Constraint (C1) uses a binary variable to ensure a serial s is only loaded on one connector v at a time.

Constraint (C2) states that the time $W_{v,k}$ at which connector v completes arc (i, j) on leg k is not less than the time at which leg $k-1$ is completed, plus transit time $T_{v,i,j}$ plus the layover time *LAYOVER*.

Constraint (C3) states that connectors v may only make one transit (i, j) per leg k .

Constraint (C4) states that the total deck space of serials being delivered by connector v on leg k from i to j cannot exceed h_v .

Constraint (C5) states that connector v can only make transit (i, j) on leg k if it also made a transit to i on leg $k-1$.

Constraint (C6) states that serial s can only be delivered by connector v on leg-transit (k, i, j) if that serial arrived at i on a different leg-transit (same or other connector).

Constraint (C7) states that the arrival time of serial s at j , $Z_{s,j}$, is not less than the connector v leg-transit (k, i, j) completion time $W_{v,k}$ plus *LAYOVER*, or zero for all trips (v, k, i, j) on which serial s does not arrive.

Constraint (C8) records that serial s that did not reach its destination e_s , if applicable.

Constraint (C9) states that the shortest-path distance (SPD) U_s from serial s final location to its intended destination e_s is not less than the SPD from its source B_s minus the sum of the differences ($SPD_i - SPD_j$) for all load transits (i, j) conducted.

Constraint (C10) states that the k^{th} leg-completion time of connector v $W_{v,k}$ is not less than the transit time $t_{v,i,j}$ plus the arrival time $Z_{s,i}$ for any cargo loaded from i to j , or zero for cargo not loaded.

Constraints (C11) and (C12) define decision variable domains.

B. SCHEDULE MIXED INTEGER PROGRAM INPUT DATA

Since a network flow problem is defined by a graph plus data, we now discuss how these are defined by the user. Major Freeman (2019) created an Excel workbook to hold these inputs, organized into the following four worksheets: WhatWhere, HowFar, ConnectorData, and ExperimentDesign, (see Appendix A: ConnectorFarmer User's Manual (V0.14) (Upton 2021)). The details of the first three are presented in this chapter, and the ExperimentDesign sheet is discussed more fully in Chapter IV. Table 1 provides an overview of each worksheet and the required information to perform an experiment and information to assist the user in tracking.

Table 1. Summary of input data specified via the Excel workbook.
Variables in red text indicate minimal requirements for the model to run

	WhatWhere (5)	HowFar (4)	ConnectorData (8)	ExperimentalDesign (5)
Variable	Site	Loc	Type	Design Point
	Type	Access - Connector X	Square Footage	Connector X Qty
	Lat	Access - Connector Y	Broken Stow Factor	Connector Y Qty
	Long	Access - Connector Z	Usable Square Footage	Connector Z Qty
	StS Site		Sust. Speed - Sea State 1	SeaState
	StS Lat		Sust. Speed - Sea State 2	Total Cap
	StS Lon		Sust. Speed - Sea State 3	Avg Spd
	Source		Sust. Speed - Sea State 4	VARLIM
	Serial Desc.			MAXLOOPS
	TAMCN			OVERLAP
	Length			mip_tolerances_mipgap
	Width			timelimit
	Weight			
	Stick Value			

Note that we use generic examples of connectors, named Connector X, Connector Y, and Connector Z. These could represent different transport platforms such as fixed-wing aircraft (e.g., C-130), rotary-wing aircraft (e.g., MV-22), or maritime surface craft (e.g., the Stern Landing Vessel (SLV) or Landing Craft Unit (LCU)).

1. WhatWhere

A snapshot of the WhatWhere sheet is given in Figure 4. Not all the columns on this sheet are required to run the model; some are simply provided for additional bookkeeping purposes. Columns not used by the model are indicated in Figure 4 with black text column headers. One example is the TAMCN column, which the user can use to track the Table of Authorized Material Control Number (TAMCN). Other columns not needed by the model, but useful for the user, are Site (which can be mapped to a specific unit), Lat/Long, ShiptoShore Lat/Long, Serial description, and Weight. Inputs required by the model are indicated by the use of red text in the column header and include Source (where the piece of equipment originates), StS Site (ship to shore site; its destination node), Length and Width, and Stick Value. Each line item in this worksheet corresponds to a “stick,” which is an individual piece of equipment. Even though in reality, weight would also be used to constrain connector loads, the current formulation considers only length and width. A heuristic within the code maps sticks to serials, and the individual serials are what get transported by the connectors. Each serial receives a weight, which is the sum of the stick values that comprise it. Higher stick/serial values indicate higher priority.

Site	Type	Lat	Lon	StS Site	StS Lat	StS Lon	Source	Serial Desc.	TAMCN	Length	Width	Weight	Stick Value
5			C				A			316	98	46400	1
5			C				A			316	98	46400	1

Figure 4. Snapshot of the WhatWhere worksheet, best viewed in color.
Adapted from Freeman (2019).

2. HowFar

The HowFar worksheet depicted in Figure 5 contains two categories of information, Connector Access (blue header) and the Distance Matrix (grey header). The Connector Access section contains a column for each connector type and the value indicates whether

or not that connector type can access each node (nodes A through H in our example). Here, for example, Connector X can access all nodes except C, Connector Y can access all nodes except D and E, and Connector Z can access all nodes except B.

The Distance Matrix section defines the travel network's graph of nodes and arcs. The matrix entries contain the distance (in miles) between node pairs. For example, assuming a connector can access nodes A, C, and G, it can travel directly from node A to nodes C or G, which are 950 miles and 1072 miles away, respectively. No connector can travel directly from A to B, D, E, or F, however, as these distance matrix entries are blank.

Connector Access				Distance Matrix						
Access - Location	Access - Connector X	Access - Connector Y	Access - Connector Z	A	B	C	D	E	F	G
A	TRUE	TRUE	TRUE			950				1072
B	TRUE	TRUE	FALSE							
C	FALSE	TRUE	TRUE				339	303	389	
D	TRUE	FALSE	TRUE					184		
E	TRUE	FALSE	TRUE						164	
F	TRUE	TRUE	TRUE							
G	TRUE	TRUE	TRUE							
H	TRUE	TRUE	TRUE							

Figure 5. Snapshot of the HowFar worksheet, best viewed in color. Adapted from Freeman (2019).

3. ConnectorData

The ConnectorData sheet contains data about the capabilities of each connector type, and a snapshot is given in Figure 6. The data for each connector type includes the square footage available on the connector to carry equipment, the broken stow factor, which defines the proportion of square footage considered usable for storage of serials, the usable square footage (= square footage * broken stow factor), and the connector sustained speed in sea states 1 through 4. Sea state measures conditions on the surface of the water that impact operations capabilities and is primarily characterized by of wave height and frequency. Sea states above 4 would preclude most surface connector operations due to safety and operational risk and therefore are not considered by the model.

Type	Square Footage	Broken Stow Factor	Usable Square Footage	Sust. Speed - Sea State 1	Sust. Speed - Sea State 2	Sust. Speed - Sea State 3	Sust. Speed - Sea State 4
Connector X	400	0.63	252	350	350	350	350
Connector Y	8440	0.63	5317.2	14.95	12.65	10.35	9.2
Connector Z	10000	0.8	8000	13.8	11.5	9.2	6.9

Figure 6. Snapshot of the ConnectorData worksheet, best viewed in color.
Adapted from Freeman (2019).

4. ExperimentDesign

The ExperimentDesign worksheet is where the user defines the different cases, or design points in experimental design terminology, to be run. A snapshot is provided in Figure 7. At a minimum, the user must define design point number, the quantity of each type of connector (can be 0 if not used for a particular run), and the sea state. Column headers in red font indicate columns of data required for the model to run. The workbook calculates and fills in the total capacity available and average speed. The remaining columns in Figure 7 are described in more detail in the next section and Chapter IV. Briefly, VARLIM, MAXLOOPS, and OVERLAP are variables defined in the code that are used to constrain problem size and the solve process. The remaining variables shown in Figure 7, the mip_tolerance_mipgap and timelimit are CPLEX parameters that define solver stopping criteria. If the VARLIM through time limit columns are not used as part of the experimental design, values will be taken from the CPLEX configuration file or the default values in the model's code, in that order.

Design Point	Connector X Qty	Connector Y Qty	Connector Z Qty	SeaState	Total Cap	Avg Spd	VARLIM	MAXLOOPS	OVERLAP	mip_tolerance_mipgap	timelimit
1	0	1	1	1	13.3172	14.375	50	0	0	0.1	86400
2	0	1	1	1	13.3172	14.375	250	0	0	0.1	86400

Figure 7. Snapshot of the ExperimentDesign worksheet, best viewed in color. Adapted from Freeman (2019).

C. SINGLE SOLVE VERSUS ROLLING HORIZON SOLVE

We now discuss the concept of solving the model via a single iteration versus a multi-iteration rolling horizon solve implementation. When solving with a single iteration,

the number of variables required, and thus computational time, can be so large as to be prohibitive. One strategy to address this issue is to break the problem into several smaller problems, and to solve each one in succession, and that is what is implemented with a rolling horizon solve. Each rolling horizon iteration represents a single step in this multi-step solution process. A graphical illustration that may be useful to compare the single solve approach to the rolling horizon approach is given in Figure 8. The horizontal axis depicts the number of travel legs, k , and the vertical axis depicts the number of steps or iterations performed via the rolling horizon solve. The light blue block represents a single solve, which requires a much larger number of variables, and therefore much more computational time, than any single iteration of the rolling horizon solve. In contrast, the rolling horizon solution process optimizes via successive iterations, where each iteration is illustrated in Figure 8 with an individual dark grey box labeled with iteration number.

VARLIM, short for variable limit, is implemented as a soft limit on the number of decision variables to allow in the model. Once this limit is passed inside the code's looping procedure, no additional connector legs will be added. Therefore, since VARLIM determines the number of legs that are included in a single iteration, the horizontal width of the blue single solve blue block represents VARLIM_s, the variable limit for a single iteration solve, and the horizontal width of an individual iteration grey box represents VARLIM_r, the variable limit for an individual iteration rolling horizon solve. With only a single step, VARLIM_s would have to be sufficiently large as to allow all serials to be delivered, but preferably, not indiscriminately large so as to avoid unnecessarily long run times. Additionally, it is important to consider the size of VARLIM_r; if the value is too small or too large the rolling horizon strategy can be ineffective.

OVERLAP is the number of previously solved connector transit legs that are eligible for re-solving in the next iteration. This allows decisions made by the solver to be revisited and therefore hopefully improved with respect to minimizing the objective function. The OVERLAP is depicted graphically in Figure 8 with the dark gray box, drawn to include variables from Iteration 1 that are also included in Iteration 2. Variables from one iteration that are *not* eligible for resolve in the next iteration are indicated by the light grey boxes labeled "Variables Fixed."

This rolling horizon process will continue until a full solution (within the specified tolerance of optimality) is found or MAXLOOPS, a hard limit on the maximum number of iterations that can be performed, is reached. Both the tolerance on optimality (mip_tolerance_mipgap) and MAXLOOPS are specified by the user. A notional full solution that has been reached is represented by a black dashed line in Figure 8. MAXLOOPS is represented by a red dashed line and represents a hard limit on the number of rolling horizon iterations, if a full solution is not found before that iteration. If a full solution is found on iteration n (where $n < \text{MAXLOOPS}$), then any iterations beyond n but less than or equal to MAXLOOPS will not be executed.

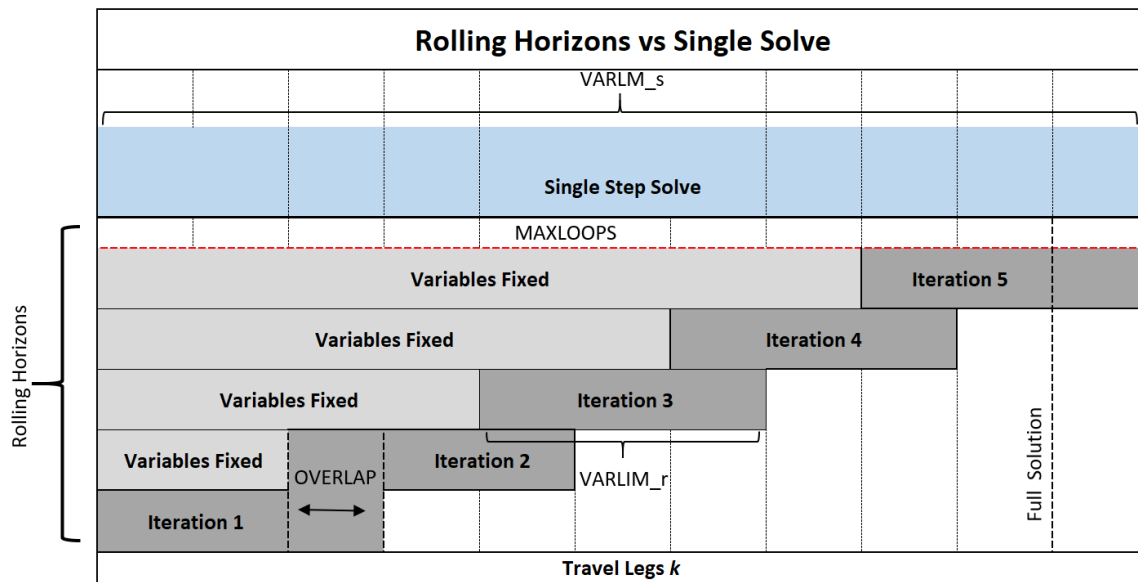


Figure 8. Rolling Horizons vs. Single Solve solution strategies, best viewed in color

In Chapter IV we explore and compare the usage of the single solve process versus the rolling horizon solve process via experimentation.

D. NPS IMPROVEMENTS TO THE SCHEDULE MIXED INTEGER PROGRAM

The Operations Analysis Directorate (OAD) of the Combat Development and Integration (CD&I) identified aspects of S-MIP that required further investigation, and a FY21 Naval Research Program (NRP) project funded these improvements.

As part of this NRP project, Professor Emily Craparo, of Operational Research Department at NPS, investigated and improved the rolling horizon implementation within the model code. Specifically, Craparo (1) fixed issues with how the code was handling serials from iteration to iteration, including initial positions and calculation of remaining distance; (2) reduced the size of the *S_TRIPS* and *S_LOADS* sets to exclude infeasible combinations, without eliminating any allowable combinations; and (3) modified the code to allow OVERLAP to change on each iteration, either upward or downward, choosing the largest possible value for OVERLAP on each iteration. The change to the OVERLAP implementation requires more calculations than the previous downward-only movement implementation, but makes full use of VARLIM each time. A full discussion of the model's code and these improvements are beyond the scope of this thesis, but further details can be requested by referring to NRP Project ID NPS-21-M133-A, "An Integrated Framework to Develop and Analyze Concepts of Employment for Surface Connectors."

An additional important improvement made to the model was to eliminate the non-determinism previously exhibited by the model. This improvement to the code was made by Steve Upton of the SEED Center. Before this fix, even though the optimization model and solution process were not intended to be stochastic given a fixed set of inputs, multiple runs of the model would yield different output (sometimes quite significantly). Basically, NPS determined that the observed non-determinism resulted from the use of unordered Python sets in the construction of the problem instance sent to the solver. The issue was resolved by replacing the use of unordered sets with the use of ordered lists and also, by switching to the use of a deterministic time limit (CPLEX setting *dettimelimit*) instead of a time limit based on system or wall clock time.

In the next chapter, we describe and analyze focused experiments to explore the impact of these changes as well as to explore running the model with one large single iteration solve process versus the default rolling horizon implementation.

IV. EXPERIMENTATION

This chapter discusses the experiments performed on S-MIP, the software used to run and post-process the experiments, design of experiments, and the analysis conducted. It also discusses metrics analyzed to capture solution quality, including proportion of serials delivered and time to deliver, and model run time.

A. SOFTWARE

Software developed by Stephen Upton of the SEED Center in 2020, called ConnectorFarmer (v0.14), was used to execute the experiments. ConnectorFarmer was developed as a data farming wrapper around OAD’s S-MIP, using a collection of Python scripts. It can use multiple cores on a single machine to run multiple instances of S-MIP concurrently. ConnectorFarmer consists of two main scripts, ConnectorRunner and ConnectorMiner. ConnectorRunner is the script used to execute the runs and ConnectorMiner is the script used to post-process the output, extracting and computing various summary metrics, and creating an Excel workbook that contains more detailed output. ConnectorFarmer can also use multiple cores on a single machine to post-process individual run output concurrently. For further information about the software, see the Connector Farmer User’s Manual in Appendix A.

B. DESIGN OF EXPERIMENTS

Models or simulations that provide decision support for complex situations usually have numerous inputs or parameters. With any model that receives inputs and generates outputs, much can be gained by exercising the model over numerous inputs to quickly gain insights into how the model is working—insights that would be difficult or impossible to obtain if we limited ourselves to a small number of ad hoc runs. By conducting efficiently designed experiments, an analyst can gain a deeper understanding of how the model behaves, uncover key drivers or sensitivities amongst the inputs, and identify interesting breakpoints or so-called “knees in the curve” of influential inputs. Additionally, the analyst may use experimentation to help identify robust strategies, that is, strategies that work well despite variation or uncertainty about model inputs. The overall goal is to aid decision

making (Kleijnen et al. 2005). The design of experiments (DOE) methodology allows us to vary the inputs we are interested in, referred to as experimental factors, to determine how they impact key outputs, or responses (Sanchez et. al. 2020). The different values each factor can take in an experiment are called levels.

Without any knowledge of DOE, one might simply try to guess, in an ad hoc manner, which levels of the factors to try. However, this would not be an effective strategy for several reasons, including (1) inputs may become highly correlated with each other, and (2) an interaction effect may be missed (Sanchez et. al. 2020). Let us now look at a simple illustration of DOE.

Figure 9 illustrates that we sampled (or tested) two factors, called speed and stealth, in order to understand their effect in a simple game of capture the flag. Each dot in the figure represents a specific combination of input factors, called a design point (DP) in DOE terminology. Two different extremes are represented in Figure 9: a sparse design space (left) and a denser design space (right). For the sparse design, we see that these two factors are evaluated at two different levels each, a notional low value and a high value. Every possible combination is tested, resulting in four design points: (1) low stealth and low speed; (2) high stealth and low speed; (3) low stealth and high speed; and (4) high stealth and high speed. Since every possible combination was tested, this design is called a full factorial design. One benefit of a full factorial design is that every possible comparison of the levels sampled can be made and every effect estimated. For a factor at two levels, only linear effects can be estimated. The denser design space on the right is also a full factorial design, but this time, each factor can take on eleven different values, as opposed to just two. This enables an analyst to fit much more complicated metamodels.

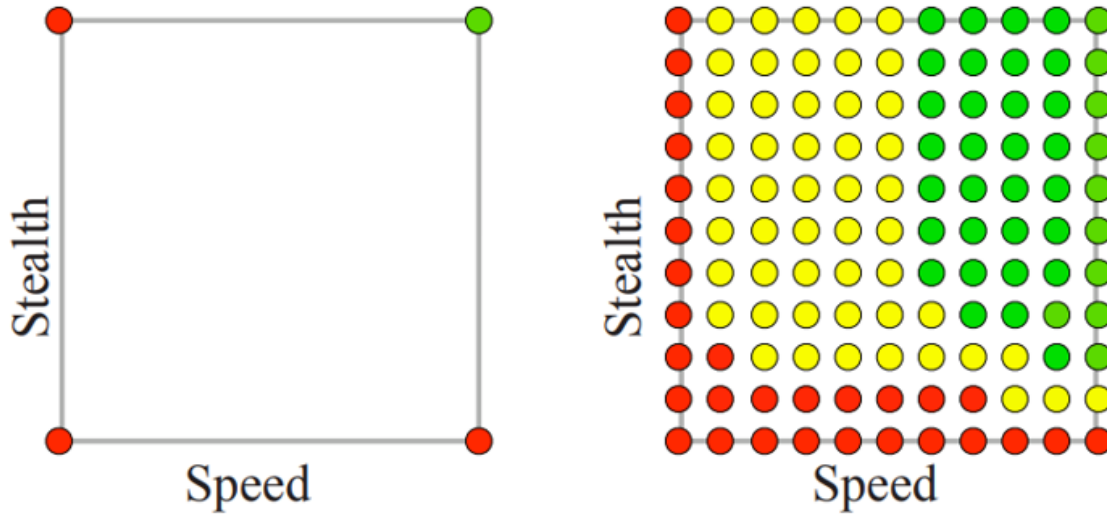


Figure 9. Capture the flag illustrating a sparse full factorial design (2^2) and a denser full factorial design (11^2). Source: Sanchez et al. (2020).

Clearly, the design on the right yields more information, but as the number of factors and number of levels for each factor increases beyond two, it is not efficient and simply may not be feasible to conduct full factorial designs. As the number of factors increases, the curse of dimensionality quickly rears its head and limits what is possible within any given time frame, given computational resources. Of course, high performance computing resources with access to hundreds or thousands of processors will help, but will not make the number of DPs an unlimited quantity. The good news, however, is that large-scale experimentation, in which many factors can be sampled over many levels, can be achieved by using efficient and flexible space-filling designs, such as a first- or second-order nearly orthogonal Latin hypercube (NOLH) (MacCalman 2017). Combined with the use of multiple processors, if available, studies that would have taken years or even lifetimes to complete become possible in hours or days.

For the experiments we conducted and next describe, we intentionally chose to start small, varying only two or three factors, and we employed the full factorial design. These experiments lay the groundwork, though, for conducting larger, space-filling designs as a follow-on effort.

C. EXPERIMENT 1

Recall that S-MIP model can be solved using the strategy of one single solve iteration or via the multi-iteration rolling horizon solve process. The first experiment uses the rolling horizon solve. Table 2 contains the description of factors and their respective levels for Experiment 1.

Table 2. Descriptions of factors and levels for Experiment 1 involving rolling horizons

Experiment 1 - ROLLING HORIZON SOLVE	
Factor Name	Levels
VARLIM	6000, 8000, 10000
OVERLAP	1, 2, 3
mip_tolerances_mipgap	.1, .15

VARLIM, OVERLAP and mip_tolerances_mipgap are varied to determine their impact on solution quality (proportion of serials delivered and time to deliver) and run time. VARLIM and OVERLAP were defined in detail in Chapter III. The mip_tolerances_mipgap factor, hereafter referred to as mipgap, is a CPLEX solver exit criterion for a full solution, and represents a relative tolerance, or user acceptance, on the gap between the best integer objective and the objective of the best node remaining. For example, for CPLEX to stop as soon as it has found a feasible integer solution within 10 percent of optimal, mipgap would be set to 0.10. Experiment 1 varies mipgap over the levels of 10% and 15%.

D. EXPERIMENT 2

Experiment 2 uses a single iteration solve, as opposed to the rolling horizon solve used in Experiment 1. To our knowledge, S-MIP model has never been executed in this manner; therefore, this experiment provides valuable insight on useful settings, such as how large VARLIM needs to be and how much time it takes to run. To accomplish a single iteration solve, the user simply sets MAXLOOPS to 0 and is then free to experiment with

different values of VARLIM to find one that is sufficiently large to ensure all serials are delivered, but no larger to avoid unnecessarily long run times. It is important to note, however, that the determination of a sufficient or “best” value for VARLIM is instance-dependent, meaning that given a different set of inputs (different equipment, different connector set, different CPLEX parameters), would likely result in a different sufficient value for VARLIM. We also vary mipgap in this experiment, including a higher level, 20 percent, than was included with the first experiment. Since we expect run times to be long, we will examine if perhaps the tolerance on optimality could be increased without sacrificing much in the way of solution quality. Table 3 contains the description of factors and their respective levels for Experiment 2. Of note is that we originally intended the VARLIM value of 20,000 to be included in this experiment, but it was clear early on that run times were going to be longer than anticipated, and additionally, an early run indicated that this value would not be close to sufficient to deliver all serials. Therefore, due to that observation and the time constraint to finish the runs for this thesis, we chose to eliminate 20,000 from further consideration.

Table 3. Descriptions of factors and levels for Experiment 2 involving single solve

Experiment 2 - SINGLE ITERATION SOLVE	
Factor Name	Levels
VARLIM	30000, 40000, 50000
mip_tolerances_mipgap	.1, .15, .2

E. HIGH PERFORMANCE COMPUTING

The runs were conducted on the SEED Center’s “reaper” high-performance computing cluster and NPS’ “Hamming” high-performance computing cluster. On the “reaper” cluster, we used one compute node configured with Windows Server 2008R2, 512 GB of RAM, and 32 processors, running from 1 to 18 simultaneous experiments. On “Hamming,” we used from 16 GB up to 240 GB of RAM, and up to 16 processors on

several Linux compute nodes, using up to 16 threads for a single process. We observed run times for a single rolling horizon solve that varied between approximately 30 and 60 hours.

F. ANALYTICAL SOFTWARE

The resulting output data from the experiments was analyzed using JMP Pro version 15—statistical software product developed by SAS Institute (www.jmp.com). JMP provides a wide variety of methods for data wrangling, analysis, and visualization. One analysis method we used is called a partition tree, which seeks to explain variability in a response (output) as a function of the factors, or predictors, through construction of a tree-like hierarchical structure. By partitioning the data recursively with this method, an analyst can gain valuable insight into the relationships between the response and a set of predictors. Partition trees are non-parametric, so there are no assumptions about the underlying data. Their structure, which might be loosely interpreted as a decision tree, is generally considered intuitive and easy to brief, even to non-technical decision makers. They also handle discontinuities or jumps in the data and interactions between predictors quite naturally, and additionally, they provide insight into key threshold values via the cut points. For larger experiments, it may be useful to complement the use of a single best tree with a random forest of trees, but for our small experiments, we used a single best tree. Of note, though, is that the random forest provides a more equitable opportunity for all predictors to explain variability, and they can also prevent overfitting. We also utilized JMP to perform ordinary least squares regression. We found that partition trees and regression complement each other nicely, due to the different strengths, limitations, and assumptions of each method.

V. RESULTS

A. COMPARISON OF RESULTS (REVISED CODE VERSUS ORIGINAL)

We started by comparing a set of runs obtained by running the original model to one run using the revised model, with all input data provided via the Excel workbook held constant. The instance run corresponded to 27 of a notional “threshold” version of a Light Amphibious Warship (LAW) traveling in sea state 1. Only one run of the revised model was made because, as mentioned in Chapter III, one of the implemented improvements was to fix the previously encountered issues with non-determinism.

Both the revised and original models were run with a VARLIM of 50,000. Preliminary runs indicated that the original default of 10,000 was too low to allow for proper use of OVERLAP, i.e., it was being forced to zero because the number of variables needed far exceeded VARLIM. Additionally, improvements provided by Professor Craparo eliminated the use of unnecessary variables, thereby allowing OVERLAP to be set to the value of two. Testing indicated that an OVERLAP of two was not possible with the original model, assuming $\text{VARLIM} = 50,000$, so its OVERLAP was set to one. In other words, we can think of this experiment as having been run with OVERLAP equal to two for each model, but we already knew from previous runs that the code will force OVERLAP to one for the original model. In contrast, the improved model allows OVERLAP to stay at two and uses that additional functionality to help improve the quality of the solution.

Figure 10 contains a snapshot of the data obtained by running ConnectorMiner. All of these runs resulted in all serials being delivered ($\text{prop_serials_delivered}$ equals 1). The time to deliver the last serial was 19.4 days with the revised model and ranged from 20.4 to 25.3 days with the original model.

Model	Rep	LAW-Th Qty	LAW-Obj Qty	SeaState	OVERLAP	VARLIM	obj_fun_value	prop_serials_delivered	time_to_deliver
ORIG	1	27	0	1	1	50000	3833.908154	1	20.5
ORIG	2	27	0	1	1	50000	3901.343846	1	22
ORIG	3	27	0	1	1	50000	5237.134692	1	25.3
ORIG	4	27	0	1	1	50000	3848.507462	1	20.4
ORIG	5	27	0	1	1	50000	3859.056923	1	20.5
REVISED	1	27	0	1	2	50000	3844.201462	1	19.4

Figure 10. Snapshot of ConnectorMiner output

Figure 11 depicts the histograms and summary statistics for individual serial delivery times for the five runs of the original model (bottom) and one run of the revised model (top). The revised model delivered the last serial at 19.4 days with a median delivery time of 9.9 days. The five runs of the original model resulted in a maximum delivery time of 25.3 days, with a median delivery time of 11.3 days. We conclude that the model improvements were not only necessary, but impactful in terms of solution quality.

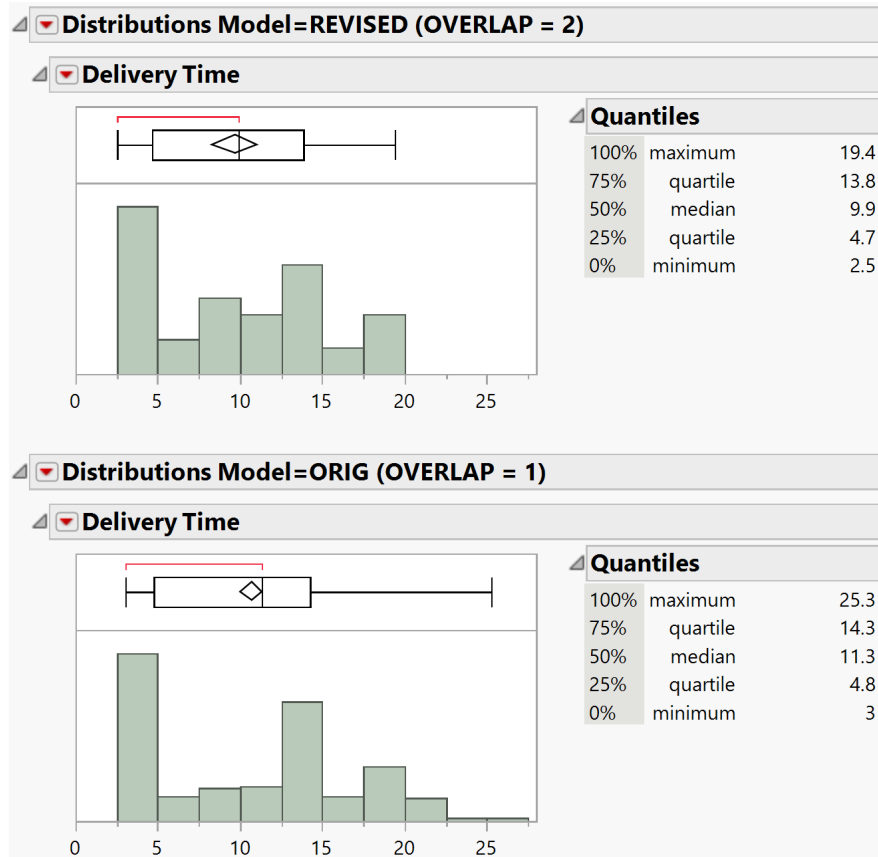


Figure 11. Histograms and summary statistics for individual serial delivery times for five runs of the original model (bottom) and one run of the revised model (top).

B. EXPERIMENT 1 (ROLLING HORIZON SOLVE)

For Experiment 1 and Experiment 2, the connector mix consisted of eight Stern Landing Vessels (SLV) traveling in sea state 1. Each of the 18 DPs of Experiment 1 resulted in all serials being delivered. A snapshot of the post-processed data appears in Figure 12. The best design points are highlighted in green, corresponding to a VARLIM of 6,000, OVERLAP equal to one, and mipgap at 0.1 or 0.15. For these two DPs, the proportion of serials delivered was 1.0 and time to deliver was 23 days. The run time ranged from approximately 31 hours to 60 hours.

Design Point	SLV Qty	SeaState	VARLIM	OVERLAP	MAXLOOPS	mip_tolerances_mipgap	obj_fn_value	prop_serials_delivered	time_to_deliver	num_steps	run_time(hrs)
1	8	1	6000	1	10	0.15	72634.5	1	23	3	44.97
2	8	1	6000	1	10	0.1	72628.7	1	23	3	45.73
3	8	1	6000	2	10	0.15	75345.9	1	26.1	7	44.99
4	8	1	6000	2	10	0.1	70013	1	23.5	7	50.82
5	8	1	6000	3	10	0.15	75345.9	1	26.1	7	43.24
6	8	1	6000	3	10	0.1	70013	1	23.5	7	51.83
7	8	1	8000	1	10	0.15	67678.2	1	29.2	4	46.66
8	8	1	8000	1	10	0.1	67620.4	1	30.4	4	44.45
9	8	1	8000	2	10	0.15	68954.4	1	25.5	4	47
10	8	1	8000	2	10	0.1	70509.6	1	25.5	4	53.21
11	8	1	8000	3	10	0.15	67315.8	1	25.5	5	54.45
12	8	1	8000	3	10	0.1	72458.2	1	23.6	5	58.45
13	8	1	10000	1	10	0.15	67623.2	1	26.1	3	31.22
14	8	1	10000	1	10	0.1	67559.4	1	26.1	3	33.55
15	8	1	10000	2	10	0.15	67306.3	1	25.5	4	47.33
16	8	1	10000	2	10	0.1	70311.9	1	25.5	4	55.64
17	8	1	10000	3	10	0.15	67602.8	1	24.4	5	55.43
18	8	1	10000	3	10	0.1	71011.2	1	24.6	5	60.17

Figure 12. Snapshot of ConnectorMiner output for Experiment 1

Since we already knew that all serials were delivered, we fit a partition tree to time to deliver all of the serials, and this is shown in Figure 13.

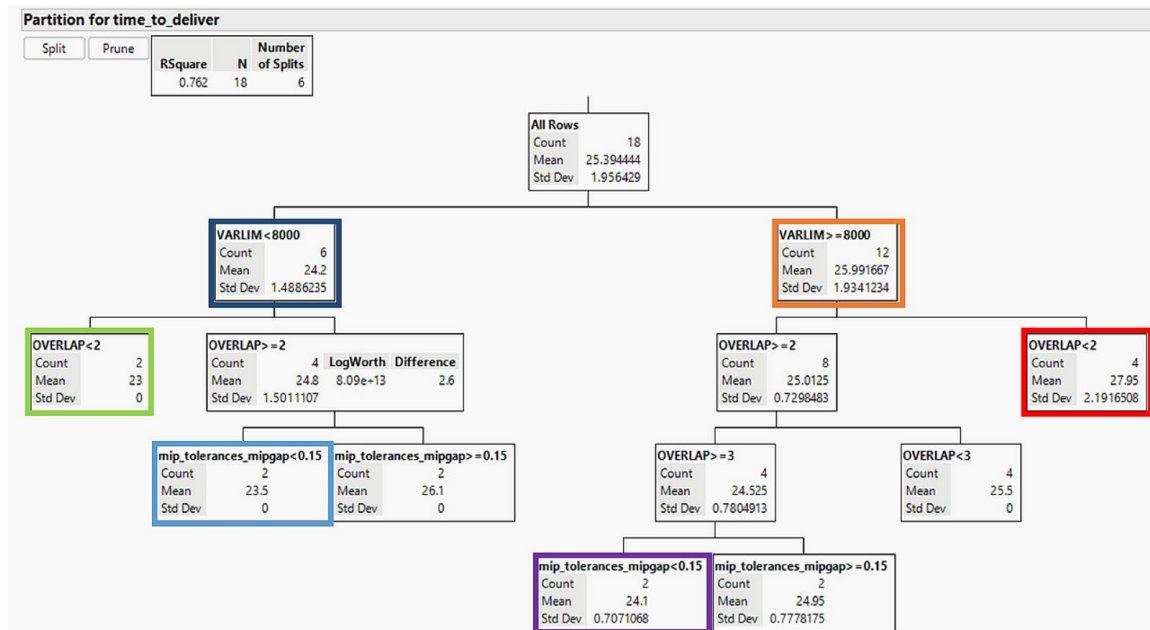


Figure 13. Partition tree for time to deliver

As shown in Figure 13, the first split occurs on VARLIM (illustrated with blue and orange boxes), and further down the left branch is the path leading to the lowest time to deliver, which was 23 days (green box). This best result was achieved with a VARLIM of

6000 (since this was the only level less than the split value of 8000) and OVERLAP of one (since this was the only level less than the split value of two). With VARLIM of 6000 and OVERLAP of two or three, a close second of 23.5 days (light blue box) was obtained if the tolerance on optimality was 10% (since 0.10 is the only level less than the split point of 0.15).

With the higher VARLIM values (orange box), a less favorable result was obtained, especially if limiting OVERLAP to one, in which case the time to deliver was almost 29 days (red box). If using an OVERLAP of three and a tolerance of 10%, a reasonably close result of 24.1 days (purple box) was achieved.

The solver status at the end of each run (not shown here) indicated that some of these DPs terminated when the user-specified deterministic time limit was exceeded, prior to reaching an integer optimal solution within tolerance, and other DPs terminated when an integer optimal solution was reached. We next fit a partition tree to solver status, and this is shown in Figure 14.

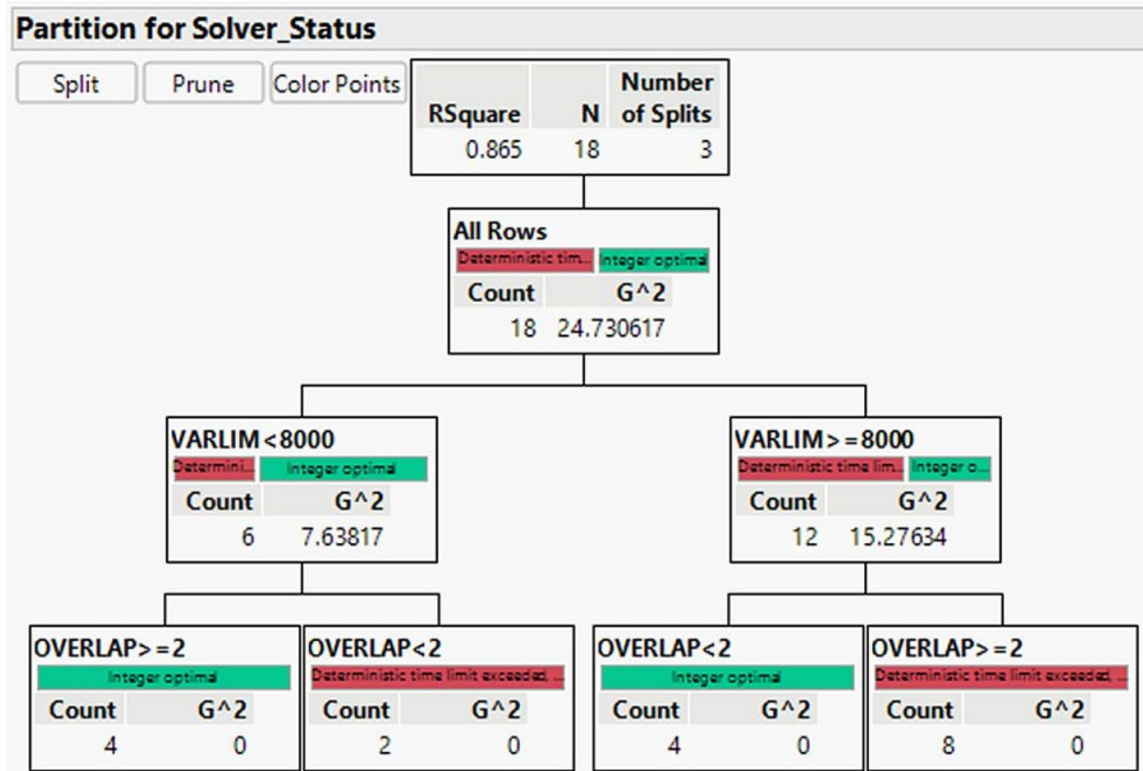


Figure 14. Partition tree for solver status

The green box around the left-most terminal node illustrates that a smaller problem, with VARLIM set to 6000, can achieve optimality when using an OVERLAP of two or three. For a larger problem, with VARLIM set to 8000 or higher, more time was needed, particularly when OVERLAP was two or three. With a higher VARLIM and OVERLAP of one, an integer optimal solution was achieved, but as we saw from the previous partition tree, the time to deliver value was not the best obtained. Of note is that a CPLEX setting that sets the deterministic time limit, in ticks per second, was not varied in the experiment. The deterministic time limit was set such that the experiment would time out at approximately 10 days. This estimate was based on previous runs, availability of processors, and time constraints for this thesis.

C. EXPERIMENT 2 (SINGLE SOLVE)

The results from the nine design points of Experiment 2 are displayed in Figure 15. The best result, highlighted in green, corresponds to a VARLIM of 40,000 and mipgap of 0.1. The proportion of serials delivered is 1.0 and the time to deliver was 21.6 days, lower than any solution from Experiment 1. We observe that run times are much higher for the single iteration solve than they were for the rolling horizon solve, the best design point took almost 233 hours to run, for the single-iteration solve experiment.

Design Point	SLV Qty	SeaState	VARLIM	MAXLOOPS	mip_tolerances_mipgap	obj_fn_value	prop_serials_delivered	time_to_deliver	num_steps	run_time(hrs)
1	8	1	30000	0	0.2	116701	0.857142857	17.9	1	233.12
2	8	1	40000	0	0.2	89198.5	0.952380952	19.6	1	204.72
3	8	1	50000	0	0.2	93129.7	1	22.2	1	216.94
4	8	1	30000	0	0.15	89283.5	0.904761905	316.5	1	212.41
5	8	1	40000	0	0.15	89519.3	0.952380952	19.6	1	176.16
6	8	1	50000	0	0.15	91264	0.952380952	21.6	1	219.72
7	8	1	30000	0	0.1	86122	0.904761905	315.6	1	228.06
8	8	1	40000	0	0.1	76819.5	1	21.6	1	232.56
9	8	1	50000	0	0.1	92147.4	0.952380952	21.7	1	196.02

Figure 15. Snapshot of ConnectorMiner output for Experiment 2

Next, we performed an ordinary least squares regression, where we consider mipgap to be nested within VARLIM. Though we were able to construct a full factorial design (using the same levels of mipgap applied for each level of VARLIM), it is more appropriate to estimate the effect of mipgap for a particular level of VARLIM, rather than across all levels of VARLIM. This is because each level of VARLIM corresponds to a different problem instance. The results are summarized in the actual by predicted plot (Figure 16) and regression plot (Figure 17) for a model with one main effect and mipgap nested within VARLIM. The actual by predicted plot (Figure 16) indicates that the model provides a good fit. With an R^2 of .93, it explains 93% of the variability in the data.

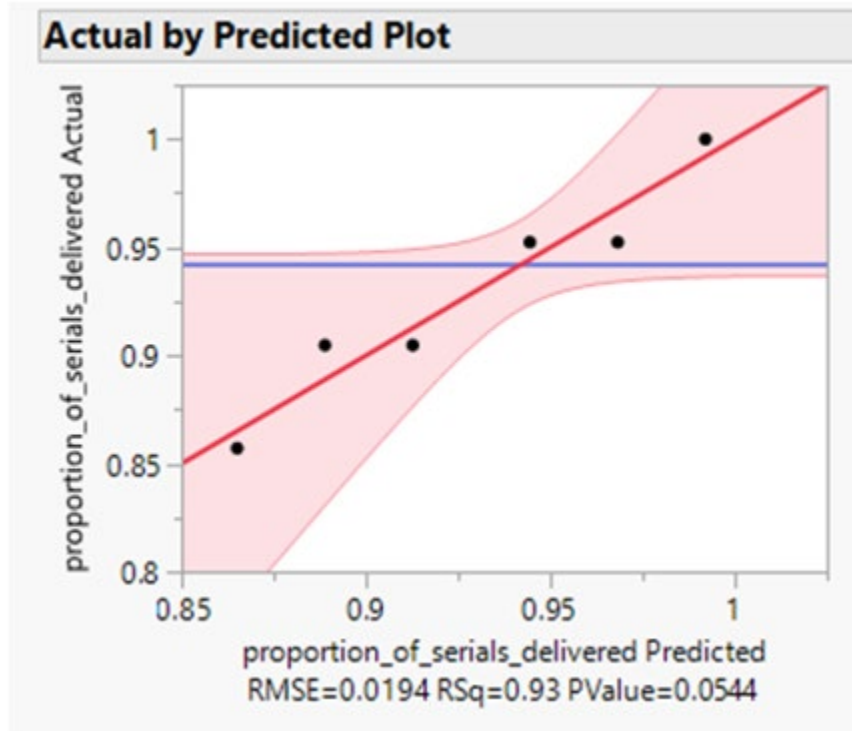


Figure 16. Actual by predicted plot for proportion of serials delivered

The regression plot in Figure 17 contains a line of fit for each VARLIM level. The first observation is that 30,000 is too low of a value for VARLIM, as none of those runs resulted in 100% serials delivered. For VARLIM of 40,000, all serials are delivered when mipgap was at its lowest setting for this experiment, 0.10. Lower mipgap is better for a VARLIM of 40,000. For VARLIM of 50,000, though, lower mipgap was actually worse than higher mipgap, presumably because with such a large problem, more computational time is needed to achieve a final solution that delivered all serials. However, when the mipgap was 0.20 and VARLIM was 50,000, all serials were delivered.

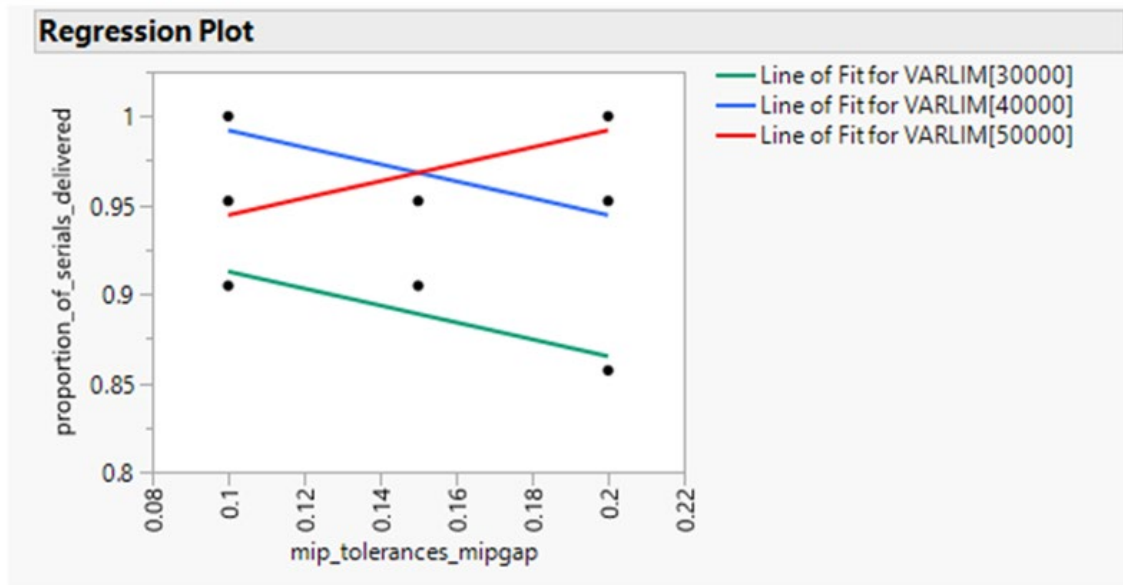


Figure 17. Regression plot for proportion of serials delivered

We next look at the model's sorted parameter estimates and prediction profiler, displayed in Figure 18. We again see that 30,000 is an insufficient value for VARLIM and that the general trend for mipgap is that lower values tend to increase the proportion of serials delivered.

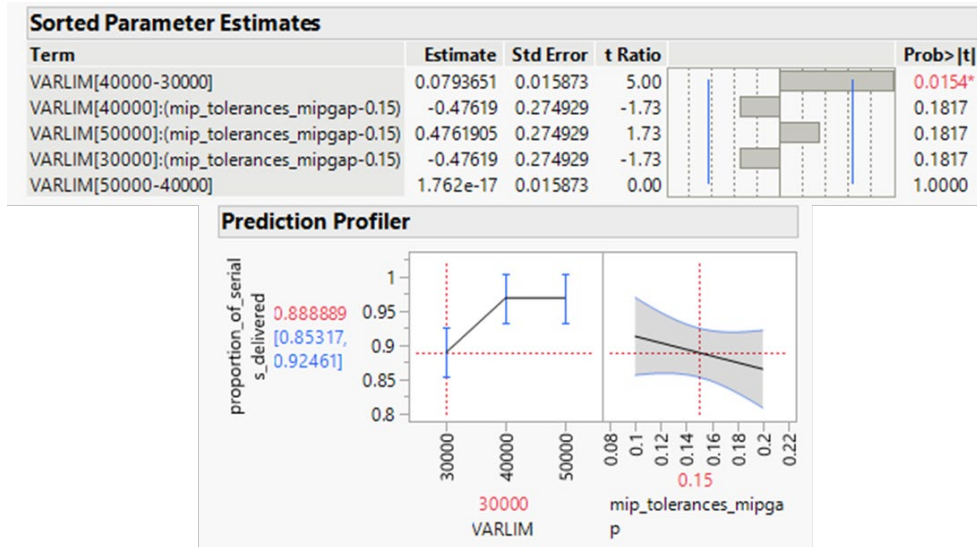


Figure 18. Sorted parameter estimates and prediction profiler generated by JMP

Finally, since we know that none of the 30,000 VARLIM runs resulted in all serials being delivered, we isolate the 40,000 and 50,000 VARLIM runs to produce the plot in Figure 19.

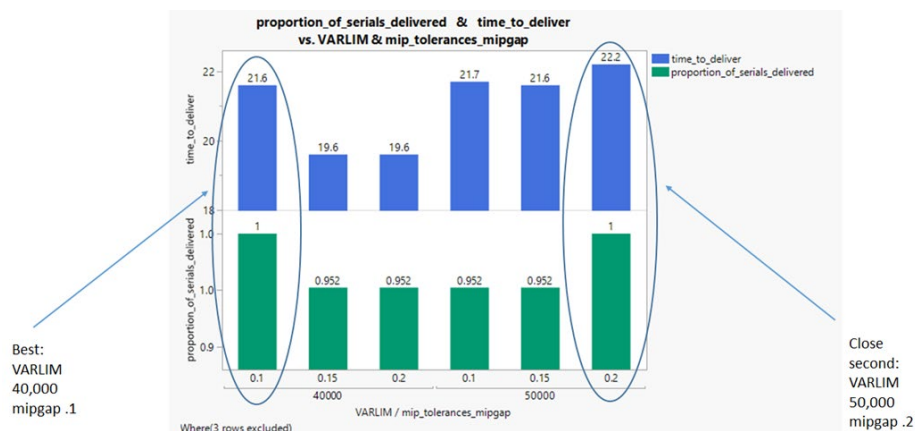


Figure 19. Proportion of serials delivered and time to deliver by VARLIM and mipgap

In Figure 19, time to deliver is illustrated with blue bars and the proportion of serials delivered with green bars. The best result, with all serials delivered within 21.6

days, was obtained with VARLIM of 40,000 and mipgap of 0.1. A close runner up, with all serials delivered within 22.2 days, was obtained with VARLIM of 50,000 and mipgap of 0.2. From this we conclude that larger values of VARLIM seem to allow for increasing mipgap without a substantial effect on solution quality.

Because this problem is so large, further improvement might be obtained by increasing the number of threads and by intentionally trading a little determinism for opportunism, which can be accomplished by setting the CPLEX parallel mode switch to either -1 (opportunistic mode) or 0 (let CPLEX decide whether to invoke deterministic or opportunistic search). All runs for this thesis were made with the parallel mode switch set to 1, the deterministic mode.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

In this chapter we discuss our conclusions from the comparison of the runs conducted with the original code versus with the improved code as well as from the experimental exploration of the rolling horizon and single iteration solution techniques. We conclude with ideas about possible follow-on research related to this thesis.

A. CONCLUSION

This thesis utilized optimization and design of experiments to gain insights into the following research questions:

- Do changes made to the original S-MIP model improve solution quality and run time?
- How might changes in the optimization solution strategy impact both solution quality and run time?
- Can data farming S-MIP provide us with a better understanding of the tool and increased confidence in its solutions?

Valuable improvements were made to the original S-MIP, including an improvement that allows for the revised model to be run with a higher value for OVERLAP than was possible with the original model, allowing for increased previous decision horizons to be revisited, using a fixed VARLIM value of 50,000. This additional functionality improved the quality of the solution, allowing for a reduction in both maximum and median times to deliver individual serials. The median and maximum serial delivery times are summarized in Figure 20. An additional significant benefit is that the improved code includes a fix for the previous issue with non-determinism in results, eliminating the need to perform multiple replications.

Best Solution Found with Revised Model Versus Original Model					
Model	VARLIM	OVERLAP	All Serials Delivered	Max Serial Delivery Time (days)	Median Serial Delivery Time (days)
Revised	50000	2	YES	19.4	9.9
Original	50000	1	YES	25.3	11.3

Figure 20. Comparison of individual serial delivery time statistics (revised model versus original model)

Using the improved model, we explored how a change in the optimization solution strategy impacts both solution quality and run time. Figure 21 provides a comparison of the best solutions found by the rolling horizon and single iteration solve runs, respectively. The best result was obtained with the single iteration solve; with all serials delivered in 21.6 days. As expected, the single iteration runs require more time to run, in this case, approximately five times longer.

Multi Iteration Rolling Horizon Versus Single Iteration Solve		
Strategy	Best time to deliver (days)	Run time (hours)
Rolling Horizons	23	44.97
Single Solve	21.6	232.56

Figure 21. Comparison of best solution and run time for the rolling horizon and single iteration runs

In conclusion, data farming S-MIP has provided a better understanding of the model and also provided a better understanding of parameters that yield good results with each solution strategy employed.

B. RECOMMENDATIONS

We recommend continued use of the improved S-MIP and continuing to experiment with the use of the single iteration solve, which produced the best solution among the runs conducted for this study. To potentially improve the solution even further while possibly reducing run time, we recommend experimenting with the use of multiple CPLEX threads and running in opportunistic mode (parallel CPLEX parameter equal to either 0 or -1).

Because the best solution strategy and CPLEX parameters are instance-specific, major changes to equipment set, connectors, or other inputs would warrant performing additional experiments before transferring to a planner for use.

C. FOLLOW-ON RESEARCH

Future research could involve a modification to the previously mentioned heuristic inside the code that maps individual pieces of equipment to serials—packages of equipment that the model moves around according to the decision variable settings. There currently is no incentive to keep equipment from the same unit together as much as possible, and this might constitute a useful variant for certain use cases.

Another option for follow-on research would be to develop a heuristic to find a feasible solution with which to “warm start” the model. Starting with a feasible solution rather than “from scratch” could greatly reduce run time and allow a planner to find a good solution more quickly, and the time savings could allow for a planner to re-run the model more frequently, to account for changes that have occurred in inputs, i.e., a connector going down, temporary closure of a port, need to eliminate an arc due to oppositional activity along that route of travel, etc.

Another possibility might be to reformulate the model to include consideration of operational risk such as the probability of a detection or kill or number and type of escorts available to protect connectors, which would allow connectors more freedom of maneuver as they deliver supplies over various travel routes.

A further possibility might be to combine the use of optimization and simulation. The combination of simulation and optimization could provide a useful hybrid, flexible approach, since simulations are better at handling complexity, operational rules at a tactical level, randomness, and dynamic interactions over time. It might be useful to use the optimization model to generate a set of connector schedules relatively close in solution quality, then simulate those schedules to evaluate and down-select based on risk due to stochastic effects and/or oppositional actions.

Finally, using data farming, larger design of experiments could be conducted that explore a wide mix of surface or air connectors needed to deliver equipment, personnel, and supplies to sustain forces during conflict.

APPENDIX: CONNECTORFARMER USER’S MANUAL (V0.14) (UPTON 2021)

Introduction.

ConnectorFarmer is a data farming wrapper around OAD’s ConnectorModel, written in python, as a collection of python scripts. It uses the multicores on a single machine to run multiple instances of the ConnectorModel concurrently. ConnectorMiner is the post-processor that extracts and computes various summary metrics from the output of a ConnectorFarmer run, and also creates an Excel output spreadsheet with intermediate results for more detailed analysis. It also can use multiple cores on a single machine to post-process individual run output concurrently.

ConnectorFarmer is composed of ConnectorRunner and ConnectorModelSEED_EMC (ConnectorModelSEED_EMC is a modification to ConnectorModelSEED, which is a modification of the original ConnectorModel). ConnectorFarmer takes as input an Excel spreadsheet formatted the same as what is used by the original ConnectorModel, where we added the capability to read in new factors from the ExperimentDesign sheet of the Excel input. These are: Broken Stow Factor and the Sustained Speed in a Sea State parameters of a connector; a set of Experiment level parameters - VARLIM, MAXLOOPS, SERIALSIZE, OVERLAP, and LAYOVER; and a set of CPLEX parameters (described in the ‘Preparing to run a designed experiment’ section). These modifications don’t interfere with the original ConnectorModel, i.e., the original ConnectorModel can read in, and use, the modified Excel spreadsheet, if desired. The original ConnectorModel ignores any ‘extra’ columns (technically, as long as the column name does not contain the string “Qty”).

When started, ConnectorFarmer:

- creates a pool of workers (using python’s multiprocessing library),
- reads in the experimental design from the ExperimentDesign sheet of the Excel input file,
- creates a run directory for each design point (DP),
- creates a DP-specific Excel spreadsheet in the run directory, and
- calls ConnectorRunner to run the ConnectorModelSEED for an individual DP instance, for each DP.

After ConnectorFarmer completes, each of the run directories will contain:

- a DP*.cplex.log file, containing the CPLEX log output for the DP,
- a DP*.log file, containing log entries from running ConnectorRunner and ConnectorModelSEED_EMC,
- a DP*.pkl file, containing the pickled Experiment object,

- a farmer-DP*.log file, containing information on what factors were changed and their settings,
- a DP*-input.xlsx file, a revised Excel input file specific for this DP,
- a DP*.barchart.jpg file, containing barchart of the items delivered in 5-day time increments, and
- a set of DP*.network_metrics_*.jpg files, containing network node and arc intensity graphics.

ConnectorMiner post-processes the output from a ConnectorFarmer run. It relies on data from the pickled Experiment object output and the log files to compute a set of summary metrics and creates an Excel workbook with several sheets containing more detailed output. After running ConnectorMiner, each of the run directories will have two added files:

- a DP*-output.csv file, which holds the set of summary metrics,
- a DP*-raw-output.xlsx file, which holds several sheets containing more detailed output.

After completing post-processing of each individual DP, ConnectorMiner then concatenates the data from all the DP*-output.csv files into a single file, merges that with the experimental design data, and then places that file into the study directory. By convention, this is named “<nameofyourstudydirectory>-alloutput.csv”.

Prerequisites.

Prior to running ConnectorFarmer and ConnectorMiner, the following software packages must be installed first. We use the Anaconda distribution (version 4.8.3), which includes the numpy/scipy, and pandas packages.

1. python (we use, and tested with, version 3.7.6)
 - a. pyomo (we use, and tested with version 5.7)
 - b. pandas (we use, and tested with version 1.1.0)
 - c. openpyxl (we use, and tested with version 3.0.3; note this comes with the Anaconda distribution but not with the generic installers downloaded from python.org)
 - d. multiprocessing (part of the python standard library)
2. CPLEX (we use, and tested with version 12.10). If CPLEX is on your PATH, then ConnectorFarmer will use that version; if you want to use a different version, then you can add a line to the cplex.config file to indicate the path to your CPLEX executable. The format is: `cplex_path:<the/path/to/your/cplex>`.

If you use the Anaconda distribution, version 4.8.3, you will need to install version 1.1.0 of pandas as we use several features that are not present in earlier versions of pandas (version 4.8.3 of Anaconda comes with pandas 1.0.1)

To do so, at the command prompt, type:

```
pip install -Iv pandas==1.1.0
```

Installation.

To install ConnectorFarmer, just unzip the ConnectorFarmer.zip file to a suitable location on your machine, such as “c:\ConnectorFarmer”. Then create a ‘studies’ folder where you put individual ConnectorModel Excel input files for your separate experiments. This ‘studies’ folder can be anywhere on your machine (preferably in a path that contains no spaces). More is explained below in the Running ConnectorFarmer section.

After unzipping, you should see these files in the directory:

- all-metrics.txt
- ConnectorFarmer.py
- ConnectorFarmerFuns.py
- ConnectorMiner.py
- ConnectorMinerFuns.py
- ConnectorModelSEED_EMCMC.py
- ConnectorRunner.py
- EAB_Inputs_Fake-test3.xlsx
- cplex.config

The Connector* files are all python scripts that are needed to run ConnectorFarmer and ConnectorMiner. The all-metrics.txt file is used by ConnectorMiner and is a line-separated list of functions to compute metrics and generate the intermediate output results in the Excel output workbooks. The EAB_Inputs_Fake-test3.xlsx file is an Excel input file that can be used for test running of ConnectorFarmer and ConnectorMiner - further described below. The cplex.config file holds the set of CPLEX parameters that can be modified.

Preparing to run a designed experiment.

Prior to running a designed experiment with ConnectorFarmer, using the underlying ConnectorModelSEED_EMCMC Mixed Integer Program (MIP), you must have an input file. This input file is in the form of an Excel workbook with four (4) sheets: WhatWhere, HowFar, ConnectorData, and ExperimentDesign. There are a number of requirements that the Excel workbook, and the corresponding sheets, must satisfy in order to start a ConnectorFarmer/ConnectorModel run. ConnectorFarmer does pre-validation of the Excel workbook to assist the user, as well as imposes additional requirements of its own to handle the additional factors permitted in the ExperimentDesign sheet.

The requirements are:

1. Each of the four sheets MUST be named as indicated, i.e., spelled exactly as above, including matching case.
2. The WhatWhere sheet MUST have columns (named exactly as is, with matching spaces and case): “Site”, “StS Site”, “Source”, “Length”, and “Width”. You can have an optional “Stick Value” column to place weight/value on each of the sticks. All other columns are ignored, so the user can place other columns with additional information useful to the user, e.g., TAMCN. “Length” and “Width” columns expect units in inches. The columns are then multiplied and divided by 144 to arrive at a square footage for the stick.
3. For the HowFar sheet, the first column MUST contain the node names (the name of the column is not used, and can therefore be anything - “Loc” is the default). There then MUST be a column for each node name, using the node name as the column name. For example, if you have a node “A” in the “Loc” (first) column, then you MUST have a corresponding “A” column. Additionally, there MUST be an “Access” column for each Connector used in the experiment. The column MUST be named “Access - CnrxX”, where CnrxX is the name of the connector. Note the space before and after the hyphen.
4. For the ConnectorData sheet, you MUST have columns named: “Type”, “Square Footage”, “Broken Stow Factor”, “Usable Square Footage”, “Sust. Speed - Sea State 1”, “Sust. Speed - Sea State 2”, “Sust. Speed - Sea State 3”, and “Sust. Speed - Sea State 4”.
5. For the ExperimentDesign sheet, you must have columns named: “Design Point” and “SeaState”. In addition, for each Connector in the experiment, you MUST have a column named “CnrxX Qty”, where CnrxX is the name of the connector (no space allowed, but hyphens are permitted).
6. If “CnrxX Qty” is in ExperimentDesign, then there MUST be a CnrxX row in ConnectorData
7. If “CnrxX Qty” is in ExperimentDesign, then there MUST be an Access column for CnrxX in HowFar, i.e., “Access- CnrxX”
8. If there are other CnrxX factors (Broken Stow, Sust. Spd) in ExperimentDesign, then there MUST be a “CnrxX Qty” in ExperimentDesign

The above requirements are the minimal that MUST be satisfied in order to start a ConnectorFarmer/ConnectorModel run. Except for the last requirement, they are implicitly documented (by the code) in the original ConnectorModel.py file, and continued in the ConnectorModelSEED EMC.py file. ConnectorFarmer does some pre-validation and checks for most of the above requirements at present, and reports failure if any of the pre-validation checks fail.

In addition to the Connector parameters that are available as part of a design, the user can also add CPLEX factors and Experiment-level factors to the design. These are added to the ExperimentDesign sheet. See the accompanying EAB_Inputs_Fake-test3.xlsx Excel workbook for examples.

Here is a list of the CPLEX factors, with their default values, that can be used in a designed experiment:

- 'threads' : 1
- 'workmem' : 16000
- 'mip_strategy_file' : 3
- 'timelimit' : 28800
- 'dettimelimit' : 216000
- 'mip_tolerances_mipgap' : 0.1
- 'mip_strategy_variableselect' : 0
- 'emphasis_memory' : yes
- 'mip_strategy_nodeselect' : 1
- 'emphasis_mip' : 1
- 'mip_strategy_branch' : 1

To use in a designed experiment, create a column with the name of the factor, spelled exactly as above (without the single quotes), and place their values in that column. If a CPLEX factor is not specified in the ExperimentDesign sheet, ConnectorFarmer will look for the factor value in a cplex.config file placed in the study directory (see the example cplex.config that comes with the ConnectorFarmer distribution). If the factor value is not found there, then it will use the default as indicated above.

You can also add five Experiment-level factors. They are, with their default values:

- 'VARLIM' : 10000
- 'LAYOVER' : 0.5
- 'MAXLOOPS' : 10
- 'OVERLAP' : 1
- 'SERIALSIZE' : None

Note that SERIALSIZE of "None" means that you are not supplying an explicit max serial size, therefore the code's SerialSizeHeuristic function will be used to determine SERIALSIZE for that run, based on the connectors available.

As with the CPLEX factors, to use in a designed experiment, create a column with the name of the factor, spelled exactly as above (without the single quotes), and place their values in that column.

For both the CPLEX and Experiment-level factors, if you misspell the name of the factor, then that factor will not change in the experiment, i.e., it will be ignored.

ConnectorFarmer documents what factors were used in the individual DP*.log files in the corresponding run directory. Here is an example from the first few lines of a DP*.log file:

```

03/26/2021 11:10:49: using CPLEX options: {'threads': 1, 'workmem': 1
6000.0, 'workdir': WindowsPath('test_v14.1/run_14'), 'logfile': Windows
Path('test_v14.1/run_14/DP14.cplex.log'), 'mip_strategy_file': 3, 'time
limit': 36000.0, 'mip_tolerances_mipgap': 0.1, 'mip_strategy_variablese
lect': 0, 'emphasis_memory': 'yes', 'mip_strategy_nodeselect': 1, 'emph
asis_mip': 1, 'mip_strategy_branch': 1, 'dettimelimit': 21600.0}
03/26/2021 11:10:49: processing Experiment top_level factors from Exp
erimentDesign: test_v14.1\run_14\DP14-input.xlsx
03/26/2021 11:10:49: found these top levels factors in the Experiment
Design: {'OVERLAP', 'VARLIM', 'MAXLOOPS'}
03/26/2021 11:10:49: using top level parameters: {'VARLIM': 10000, 'L
AYOVER': 0.5, 'MAXLOOPS': 1, 'OVERLAP': 1, 'SERIALSIZE': 'None'}

```

Once you have an Excel input file properly constructed, you are ready to run ConnectorFarmer!

Preparing to run.

As described above, the ConnectorFarmer software consists of two components: ConnectorFarmer itself, and ConnectorMiner. Both are run individually, i.e., ConnectorFarmer doesn't depend on ConnectorMiner, and vice versa. However, ConnectorMiner does rely on certain conventions used in ConnectorFarmer, e.g., the naming of the run directories, and the pickled output files rely on code in the ConnectorFarmerSEED EMC file to establish structure of the pickled Experiment objects. Both ConnectorFarmer and ConnectorMiner are designed to be run from the command line.

Running ConnectorFarmer.

As mentioned above, ConnectorFarmer is run from the command line. ConnectorFarmer takes two required arguments:

- the absolute or relative path to the Excel spreadsheet input file
- the number of processors to use for this run

If the user only supplies the two required arguments, ConnectorFarmer will run all design points in the ExperimentDesign sheet of the Excel input file. If the user only wants to run specific design points, the user can supply a space-separated list of the design points they wish to run after the first two required arguments. The design points can be names (no spaces or hyphens) in addition to numbers; e.g., "scen14". Examples below.

First, to run ConnectorFarmer, you need to change directories to where you installed ConnectorFarmer, using your preferred shell, e.g., DOS, powershell, or the anaconda python prompt. In the examples below, we assume you are using the anaconda python

prompt and that you installed ConnectorFarmer to 'C:\ConnectorFarmer' - adjust your path accordingly.

Here is the general command line invocation of ConnectorFarmer (from the directory where the ConnectorFarmer.py file resides):

```
prompt>python ConnectorFarmer.py <pathtoExcelInputfile> <numprocessors> [DPs*]
```

where:

- <pathtoExcelInputfile> is the absolute or relative path to the Excel spreadsheet input file (we assume no spaces in input file name),
- <numprocessors> is the number of processors to use for this run,
- [DPs*] is an optional space-separated list of design points (DPs) (as listed in the Design Point column of the ExperimentDesign sheet).

We recommend creating separate “study” folders, one for each designed experiment. ConnectorFarmer uses the path of the Excel input file as the study directory, and will overwrite any existing output.

Test Running ConnectorFarmer.

To test, we will use the EAB_Inputs_Fake-test3.xlsx file that accompanies this distribution as a sample input file.

- create a 'C:\connectorFarmer\studies\test1' directory,
- copy EAB_Inputs_Fake-test3.xlsx to that directory (no spaces in file name),
- (optional) copy the cplex.config file from the ConnectorFarmer directory to your study directory and make any desired changes,
- change directories to C:\ConnectorFarmer (or wherever you placed the set of Connector* files)
- start ConnectorFarmer:

```
(base) PS c:\ConnectorFarmer> python ConnectorFarmer.py c:\ConnectorFarmer\studies\test1\EAB_Inputs_Fake-test3.xlsx 2
```

Once started, in the console/command prompt window, you should start to see scrolling lines of text. The logger in ConnectorFarmer dumps its output to the console in addition to a “farmer.log” file in the study directory. There are also individual loggers for each DP and they each dump their output to their individual run directories (by convention, these run directories are created as “run_X” where X is the DP name in the study directory).

After ConnectorFarmer completes successfully, you should have two “run_X” directories in c:\ConnectorFarmer\studies\test1\, where X runs is 14 and 15 (these are read in from the Design Point column of the Excel input file). In each of those folders, you

should have ten files as described above, i.e., three DP* log files, a DP*.pkl file, an Excel input file specific for that DP, a barchart.jpg file, and four DP*.network_metrics*.jpg files (one for each of the network nodes).

Running ConnectorMiner.

As mentioned, ConnectorMiner is run from the command line. ConnectorMiner takes three required arguments:

- the absolute or relative path to the Excel spreadsheet input file (no spaces in file name),
- the absolute or relative path to the metrics file,
- the number of processors to use for this run

If the user only supplies the three required arguments, ConnectorMiner will post-process all design points in the ExperimentDesign sheet of the Excel input file. If the user only wants to post-process specific design points, the user can supply a space-separated list of the design points they wish to post-process after the first two required arguments. Examples below.

First, to run ConnectorMiner, you need to change directories to where you installed ConnectorFarmer, using your preferred shell, e.g., DOS, powershell, or the anaconda python prompt. In the examples below, we assume you are using the anaconda python prompt and that you installed ConnectorFarmer to 'C:\ConnectorFarmer' - adjust your path accordingly.

Here is the general command line invocation of ConnectorMiner (from the directory where the ConnectorFarmer.py and ConnectorMiner.py files reside):

```
prompt>python ConnectorMiner.py <pathtoExcelInputfile> <pathtoMetricsfile> <numprocessors> [DPs*]
```

where:

- <pathtoExcelInputfile> is the absolute or relative path to the Excel spreadsheet input file (no spaces in file name),
- <pathtoMetricsfile> is the absolute or relative path to the metrics file,
- <numprocessors> is the number of processors to use for this run,
- [DPs*] is an optional space-separated list of design points (DPs).

For the <pathtoMetricsfile>, you can edit the provided "all-metrics.txt" file and put in your study directory, or leave as-is. It lists all the available metrics and output sheets that can be generated by ConnectorMiner. Metrics that are outputted to the DP*-output.csv file all start with the word "compute". Metrics that create sheets in the DP*-raw-output.xlsx Excel workbook all start with the word "sheet". The code for both the 'compute_' and the 'sheet_' functions can be found in the

ConnectorMinerFuns.py file. Lines that do not start with either “compute” or “sheet” are ignored. You can intersperse comments in the metrics file, if desired (conventionally, these start with “#”). Also, the order of the “compute” metrics in the metrics file defines the order they appear in the DP*-output.csv file (duplicate entries are ignored; the first metric encountered defines the order of appearance).

Test Running ConnectorMiner.

To test running ConnectorMiner, you will need some output. We will assume you tested running ConnectorFarmer as above and have a ‘C:\connectorFarmer\studies\test1’ study directory with all the “run_X” sub-directories, and the ‘EAB_Inputs_Fake-test3.xlsx’ is in that study directory.

Start ConnectorMiner with the following command in the command window, from your ConnectorFarmer install directory:

```
(base) PS c:\ConnectorFarmer> python ConnectorMiner.py c:\ConnectorFarmer\studies\test1\EAB_Inputs_Fake-test3.xlsx all-metrics.txt 1
```

Once started, in the console/command prompt window, you should start to see scrolling lines of text. The logger in ConnectorMiner dumps its output to the console. If you use more than one processor, the output lines of text will intermix with output from processing another DP.

After ConnectorMiner completes successfully, in those two “run_X” directories in c:\ConnectorFarmer\studies\test1\, you should now have two additional output files as described above, i.e., a csv output file, and an Excel output file. In the test1 folder you should also have a “test1-alloutput.csv” file with the concatenated results from the individual DP*-output.csv files.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Burgos A and Mclean GD (2019) Optimizing maritime prepositioning force selection of ship class to respond to humanitarian assistance and disaster relief operations in the Pacific theater. Master's thesis, Graduate School of Business and Public Policy, Naval Postgraduate School, Monterey, CA. https://calhoun.nps.edu/bitstream/handle/10945/61224/18Dec_Mclean_Burgos.pdf?sequence=1&isAllowed=y.
- Christafore RM (2017) Generating ship-to-shore bulk fuel delivery schedules for the Marine expeditionary unit. Master's thesis, Graduate School Operations Research, Naval Postgraduate School, Monterey, CA.
- Concepts Branch, Marine Corps Warfighting Laboratory/Futures Directorate (2016) Littoral Operations in a contested environment: Developing a new naval operating concept. *Marine Corps Gazette* 100(2), <https://search-proquest-com.libproxy.nps.edu/docview/1767084133?pq-origsite=primo>.
- DeMarco JFC (2021) Map of South China Sea economic exclusion zones information provided to author via personal communication, April 6.
- Department of the Navy (2018) Marine Corps Doctrinal Publication (MCDP) 4, Logistics. Washington, DC, <https://www.marines.mil/News/Publications/MCPPEL/Electronic-Library-Display/Article/899840/mcdp-4/>.
- Department of the Navy (2019) Commandant's Planning Guidance 38th Commandant of the Marine Corps, Washington, DC, <https://www.marines.mil/News/Publications/MCPPEL/Electronic-Library-Display/Article/1907265/38th-commandants-planning-guidance-cpg/>.
- Department of the Navy (2021) *Tentative Manual for Expeditionary Advanced Base Operations*. Washington, DC.
- Department of the Navy (2006) Seabasing Logistics Enabling Concept. Washington, DC, <https://www.mcwl.marines.mil/Portals/34/Documents/Concepts/Seabasing%20Logistics%20Enabling%20Concept.pdf?ver=2015-12-09-124726-227>.
- Freeman N (2019) Schedule Mixed Integer Program formulation, Python code, and associated Excel worksheets, provided to the SEED Center via personal communication and shared with author December 12.

- Gardner MJ (2015) Investigating the naval logistics role in humanitarian assistance activities. Master's thesis, Graduate School of Business and Public Policy, Naval Postgraduate School, Monterey, CA. https://calhoun.nps.edu/bitstream/handle/10945/61224/18Dec_Mclean_Burgos.pdf?sequence=1&isAllowed=y.
- The Heritage Foundation (2019) Logistics: The lifeblood of military power. Accessed January 21, 2021, <https://www.heritage.org/military-strength-topical-essays/2019-essays/logistics-the-lifblood-military-power>
- IBM (2004) CPLEX Optimization Studio <https://www.ibm.com/products/ilog-cplex-optimization-studio>.
- Kleijnen JPC, Sanchez SM, Lucas TW, Cioppa TM (2005) State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing* 17(3), <https://pubsonline-informs-org.libproxy.nps.edu/doi/10.1287/ijoc.1050.0136>.
- Lucas, T.W., W.D. Kelton, S.M. Sanchez, P.J. Sanchez, and B.L. Anderson, "Changing the Paradigm: Simulation, Often the Method of First Resort," *Naval Research Logistics*, 62(4), 2015, 293–303.
- MacCalman, A., H. Vieira, and T. Lucas. 2017. "Second Order Nearly Orthogonal Latin Hypercubes for Exploring Complex Stochastic Simulations." *Journal of Simulation*, 11(2):137–150.
- Mckavitt TP (2019) An analysis of the Marine Corps depot-level maintenance budgeting model. Master's thesis, Department of Operations Research Naval Postgraduate School, Monterey, CA.
- Naval History and Heritage Command (1950) 80-G-420027 Inchon invasion, September 1950. Accessed January 23, 2021 <https://www.history.navy.mil/content/history/nhhc/our-collections/photography/numerical-list-of-images/nara-series/80-g/80-G-420000/80-G-420027.html>.
- Naval Postgraduate School Simulation Experiments and Efficient Designs (SEED) Center for Data Farming. Accessed January 21, 2021, <https://harvest.nps.edu>
- Sanchez, S. M., P. J. Sanchez, and H. Wan. 2020. "Work Smarter, Not Harder: A Tutorial on Designing and Conducting." *Proceedings of the 2018 Winter Simulation Conference*, 237–251.
- "Simulation Experiments." In *Proceedings of the 2020 Winter Simulation Conference*, edited by K. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesin. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

- Upton S (2021) ConnectorFarmer User's Manual version 0.14, unpublished user's manual, SEED Center for Data Farming, Monterey CA.
- U.S. Department of Defense (2005) Seabasing Joint Integrating Concept. Washington, DC. [https://www.jcs.mil/Portals/36/Documents/Doctrine/concepts/seabasing.pdf?ver=2017-12-28-162032-087#:~:text=Seabasing%20is%20defined%20as%20the,within%20the%20Joint%20Operations%20Area%20\(](https://www.jcs.mil/Portals/36/Documents/Doctrine/concepts/seabasing.pdf?ver=2017-12-28-162032-087#:~:text=Seabasing%20is%20defined%20as%20the,within%20the%20Joint%20Operations%20Area%20()
- U.S. Department of Defense (2021) *Department of Defense Dictionary of Military and Associated Terms*. Washington, DC. <https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/dictionary.pdf?ver=idnWjT-PxzWCi3IHTV1-xQ%3d%3d>.
- U.S. Department of Defense (2019) *Joint Logistics*. JP 4-0, Washington, DC. https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp4_0ch1.pdf?ver=2020-07-20-083800-823.
- Vego M (2015) On Littoral Warfare. *Naval War College Review* Spring 2015, Vol. 68. No 2 <http://libproxy.nps.edu/login?url=https://www-proquest-com.libproxy.nps.edu/scholarly-journals/on-littoral-warfare/docview/1660144761/se-2?accountid=12702>.
- Ward PW (2008) Optimizing ship-to-shore movement for Hospital Ship humanitarian assistance operations. Master's thesis, Department of Operations Research, Naval Postgraduate School, Monterey, CA.
- Zwolensky Todd (2007) Logistics over the shore: A review of Operation Chromite, Operation Bluebat and its relevance to today. Thesis, Military History, U.S Army Command and General Staff College, Fort Leavenworth KS. <https://apps.dtic.mil/dtic/tr/fulltext/u2/a475691.pdf>.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

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