Multiobjective Collaborative Optimization of Systems of Systems

by

Robert A. Wolf

B.S. Systems Engineering
United States Naval Academy, 1995

Submitted to the Department of Ocean Engineering and the Engineering Systems Division
in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer

and

Master of Science in Engineering Systems

at the
Massachusetts Institute of Technology
June 2005

Signature of Author: …….
Department of Ocean Engineering
May 9, 2005

Certified by: …….
Dr. Timothy J. McCoy, Associate Professor of the Practice of Naval Construction and Engineering
Thesis Supervisor

Certified by: …….
Dr. Olivier de Weck, Assistant Professor Aeronautics & Astronautics and Engineering Systems
Thesis Reader

Accepted by: ……
Dr. Michael Triantafyllou, Professor of Mechanical and Ocean Engineering
Chair, Departmental Committee on Graduate Students

Accepted by: ……
Dr. Richard de Neufville, Professor of Engineering Systems
Chair, Engineering System Division Education Committee

Page Intentionally Left Blank

Multiobjective Collaborative Optimization of Systems of Systems

by

Robert A. Wolf

Submitted to the Department of Ocean Engineering and the Engineering Systems Division
on May 9, 2005 in Partial Fulfillment of the Requirements for the Degrees of
Naval Engineer
and
Master of Science in Engineering Systems

ABSTRACT

Concept studies for warship designs typically focus on ship performance characteristics by setting design goals for such things as speed, range, and cost. However, warships generally operate as part of a larger battle or strike group. Therefore, the designs should be evaluated as part of a system of multiple ship systems since designing each ship individually may result in underutilized and excess equipment and capability; in other words an inefficient design of the system of systems.

This thesis examines the simultaneous design of several ships using the sea base concept as an example application of a network of ships working together. The number and characteristics of these ships determine the mission performance of the sea base. To properly design any of the sea base ships, the interrelationships must be included. A mission simulation is used to combine the performance characteristics of different ship designs into a single performance objective: the time to deliver a brigade size combat force to its assigned objectives.

To enable the design of multiple ships, collaborative optimization, a multilevel optimization approach, was used to decompose the problem into individual ship design optimizations with system level interfaces controlled by a system of systems optimization algorithm. This allowed each ship to use techniques and algorithms best suited to reach an optimal design without impacting the design approaches used by the other ships. The classical collaborative optimization approach was relaxed to include multiple objectives such as performance and cost, thus developing a range of solutions which represent the tradeoff between these objectives.

Thesis Supervisor: Dr. Timothy J. McCoy
Title: Associate Professor of the Practice of Naval Construction and Engineering

Thesis Reader: Dr. Olivier de Weck
Title: Assistant Professor Aeronautics & Astronautics and Engineering Systems

Page Intentionally Left Blank

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Professor McCoy and Professor de Weck for their assistance and guidance in this research endeavor. Additional thanks go out to fellow MIT students Andy Gold, Edward West, and Andy Johnson in developing the mission simulation model and ITS ship model; and to Brian Miller and Georgios Gougoulidis for providing data on air cushioned vehicles. Most of all I would like to thank my wife Amy and son Nicholas for allowing me to stay focused.

Page Intentionally Left Blank

TABLE OF CONTENTS

LIST OF FIGURES

Page Intentionally Left Blank

LIST OF TABLES

Page Intentionally Left Blank

NOMENCLATURE

Naval Architecture Nomenclature

B	ship beam
BM	metacentric radius
C_b	ship block coefficient
C_{bD}	ship block coefficient measured at the depth
C_{bT}	ship block coefficient measured at the draft
C_{IT}	transverse inertia coefficient
C_w	waterplane coefficient
D	ship depth
KB	vertical center of buoyancy
KG	vertical center of gravity
L	ship length
T	ship draft
Δ	ship displacement
ν	specific volume of seawater (35 ft^3/LT)

Optimization Nomenclature

$g(\mathbf{X})$	inequality constraint
$h(\mathbf{X})$	equality constraint
J	objective or fitness function
J_f	fitness value (objective value adjusted for constraints)
J_o	initial objective value before adjustment for constraints
J_{sys}	system objective function
k	cost-to-displacement factor
N	number of platforms
$u(\mathbf{X})$	utility function
w^i	constraint scaling factor for subsystem i
X_i	i^{th} design variable
$\mathbf{X^i}$	vector of design variables X controlled by subsystem i
$X_{L,i}$	lower bound of variable X_i
$X_{U,i}$	upper bounds of variable X_i
X^*_i	optimal or non-dominated solution
Y^*	system compatibility variables defined at the system level
Y^	system compatibility variable used at the subsystem level
Z^*	system level variable defined at the system level
$Z^\wedge$	system level variable used at the subsystem level
λ_i	i^{th} weighting factor
$\mathbf{\Omega}$	set of **X** representing the feasible set
$\Re^+$	set of positive real numbers

Note: **Bold** indicates a vector or set of multiple values throughout the thesis

Page Intentionally Left Blank

CHAPTER 1: INTRODUCTION

Many Naval concept studies for ship design focus on missions and operations to be performed by a single ship. Typically, design goals are established for things such as speed, range, and cost. However, these design goals are merely proxies for the ship's mission performance. Furthermore, in the U.S. Navy, ships frequently operate as part of a larger battle or strike group. This trend toward interdependence between ships is likely to continue and would indicate that ship design should be studied as part of a supersystem of multiple ships or system of ship systems.

A simple example of ships which must operate together to achieve success is the sea base concept. A sea base is not a single ship or ship type, but rather a network of many different ships working together to accomplish a mission: to assemble, deliver, sustain and reconstitute a joint force to and from a land objective from the sea. Each ship has its own design goals and constraints to achieve some level of performance; however, it is only a single system. The true performance is at the System of Systems (SoS) level.

To properly design any of these ships, first, the potential architectures must be understood and how each of the various systems combines into a SoS. With these architectures in mind, the entire SoS can be viewed as a whole to develop its own Mission Performance Parameters (MPP) which combines the various individual ship design MPPs. The difficulty lies in the simultaneous analysis and design of each of these individual ships which have their own multiple design goals and constraints while taking into account the interrelationships necessary to achieve an improved overall SoS design.

Systems-of-Systems

A general definition for a system is a collection of elements which, working together, produce a result not achievable by the elements alone (Maier & Rechtin, 2002). By this definition, almost anything can be considered a system consisting of subsystems which can themselves be systems, thus making the system a SoS. Therefore, formally, a SoS has no distinguishing characteristic. However, it is useful to distinguish between systems made up of subsystems and SoS made up of independent systems. The key difference is the subsystems require each other while systems within a SoS synergize with each other. For example, an airplane consists of wings, engine, and other subsystems which all require each other to cause the airplane to fly. In contrast, an airplane flying passengers across the country is part of a greater air traffic SoS which includes several aircraft, the airports, air traffic control, etc. which can all operate without the others but synergize when they operate together. Therefore, a separate definition of a SoS is useful to differentiate large and distributed systems from less complex and compact systems (Maier, 1998).

An SoS can be defined as an assemblage of components which individually are regarded as systems and which are both operationally and managerially independent. Operational independence implies that if the SoS is disassembled into its individual systems, these systems must be able to usefully operate independently to fulfill a customer purpose (Maier, 1998). For instance, an airplane engine is not operationally independent because while it provides thrust or power, only when combined with other elements (i.e. wings, fuselage, etc.) is it able to achieve a useful customer purpose (i.e. flying). On the other hand, the airplane (consisting of several subsystems) is an operationally independent system since it is able to takeoff, fly, and land without any other system, and thus could be combined with other systems to create a SoS.

Managerial independence implies that the individual systems not only can operate independently but do actually operate independently. The systems are integrated but maintain an operational existence independent of the SoS. In other words, the systems are managed in part for their own purpose rather than strictly the purposes of the whole (Maier, 1998).

Figure 1 illustrates the framework of a SoS. Notice that the subsystems of system 3 all fall within system 3 implying they are a integral part of system 3 and do not have operational independence. On the other hand, systems 1, 2, and 3 are located both inside and outside of the SoS. These systems each have their own independent design requirements. This is meant to imply that the SoS is made up of these three systems but the systems are both managerially and operationally independent.

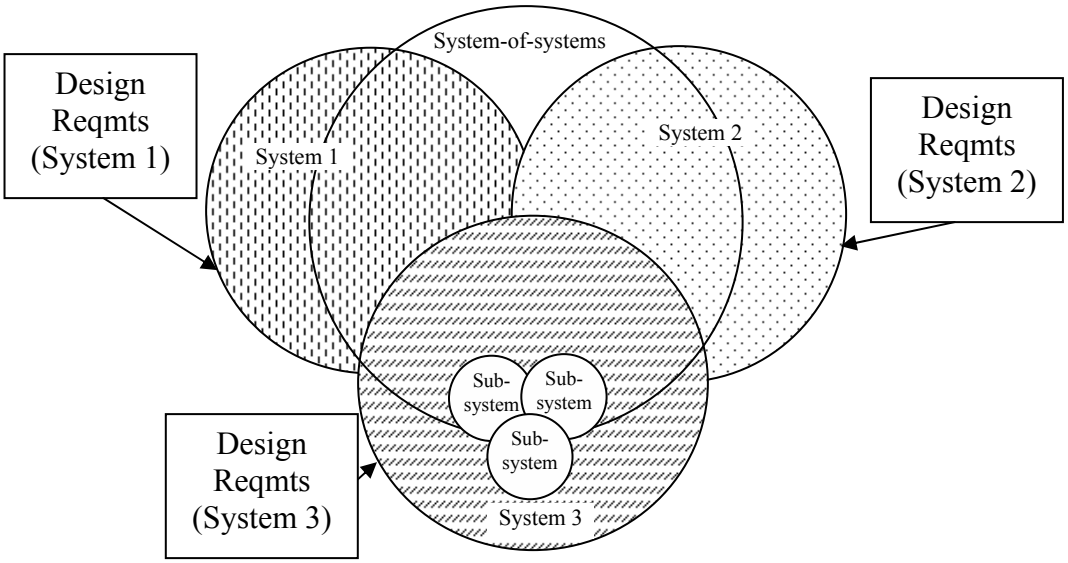


Figure 1. System of Systems Framework Showing Functional Overlap With System Independence

A SoS is more costly than a single system since it fulfills not only the purpose of the SoS but also additional purposes of the independent elements. Therefore a single system could be designed without the additional functionality for presumptively less cost (Maier, 1998). With independence comes some redundancy which optimization can be used to reduce or eliminate.

There are also other characteristics typical of SoS. A SoS is almost always complex, involving many elements with significant interactions. However, these interactions are not usually directly coupled like the airplane engine and its structure, but instead are frequently time or rule dependent. Additionally, these interactions are in many cases nonlinear and discontinuous. SoS also involve a greater amount of human involvement and thus add additional layers of uncertainty and ambiguity. Finally, the design of a SoS is typically not controlled by a single entity as might be the case for an airplane design, but rather is made up of different companies, organizations, and stakeholders without a single controlling authority.

Even with these difficulties, the impact of an individual system on the overall SoS represents part of the true utility of the individual system. In much the same way as separately optimizing subsystems on an airplane can lead to an overall suboptimal design, an optimal airplane may not achieve the desired effectiveness when operated within the larger SoS.

Defense Integrated Architectures

The United States Defense Department is one organization that has identified the need to approach design from the SoS standpoint. It has codified its approach to SoS within the Joint Capabilities Integration and Development System (JCIDS). The JCIDS process identifies "capability gaps and capability redundancies, assess the risk and priority of the gaps, and recommends the best approach or combination of approaches to address the gap" (Chairman, Joint Chiefs of Staff [CJCS], 2003, p.A-6). The process is a top down exploration of needed capabilities starting with an examination of national strategy which ultimately leads to Joint Operations Concepts (JOpsC). The JOpsC portrays "the linkage between how the joint force operates today and the vision for the future" (CJCS, 2003, p.A-3). Supporting the JOpsC are the Joint Operating Concepts (JOC) and Joint Functional Concepts (JFC). The JOCs describe how

to conduct military operations to accomplish a mission whereas JFCs describe how forces will perform a particular function. In other words, JFCs describe the capabilities while JOCs describe the operational context (CJCS, 2004a).

From the JFCs and JOCs, integrated architectures are developed. The integrated architectures are views or perspectives of Families-of-Systems (FoS) or SoS that facilitate integration, promote interoperability across, and compatibility among related architectures. Using the integrated architectures, further analysis explores the difference between "what we have and what we need to do" to define capability gaps. These capabilities should meet the following criteria:

- General enough so as not to prejudice decisions in favor of a particular means of implementation.
- Specific enough to evaluate alternative approaches to implement the capability.
- Contain the following elements: key attributes with appropriate measures of effectiveness, supportability, time, distance, effect (including scale) and obstacles to be overcome.

Following the capability analysis is a review of alternatives to satisfy the capability gaps. Through the exploration of science and technology as well as experimentation, various alternatives can be assessed both in terms of risk, cost, and effect to determine the best solution available. The entire process is illustrated in Figure 2 (CJCS, 2003).

There are two main points about the JCIDS process. First, as mentioned before, it begins with a solution neutral need and then explores possible methods to meet that need. Second, integrated architectures must look at interrelationships among several different systems without being system specific. The goal is to be able to use the integrated architecture to help identify what additional systems are needed and how they should operate with existing systems. To

make this system selection requires optimization to ensure that the nation gets the best possible capability for its limited resources.

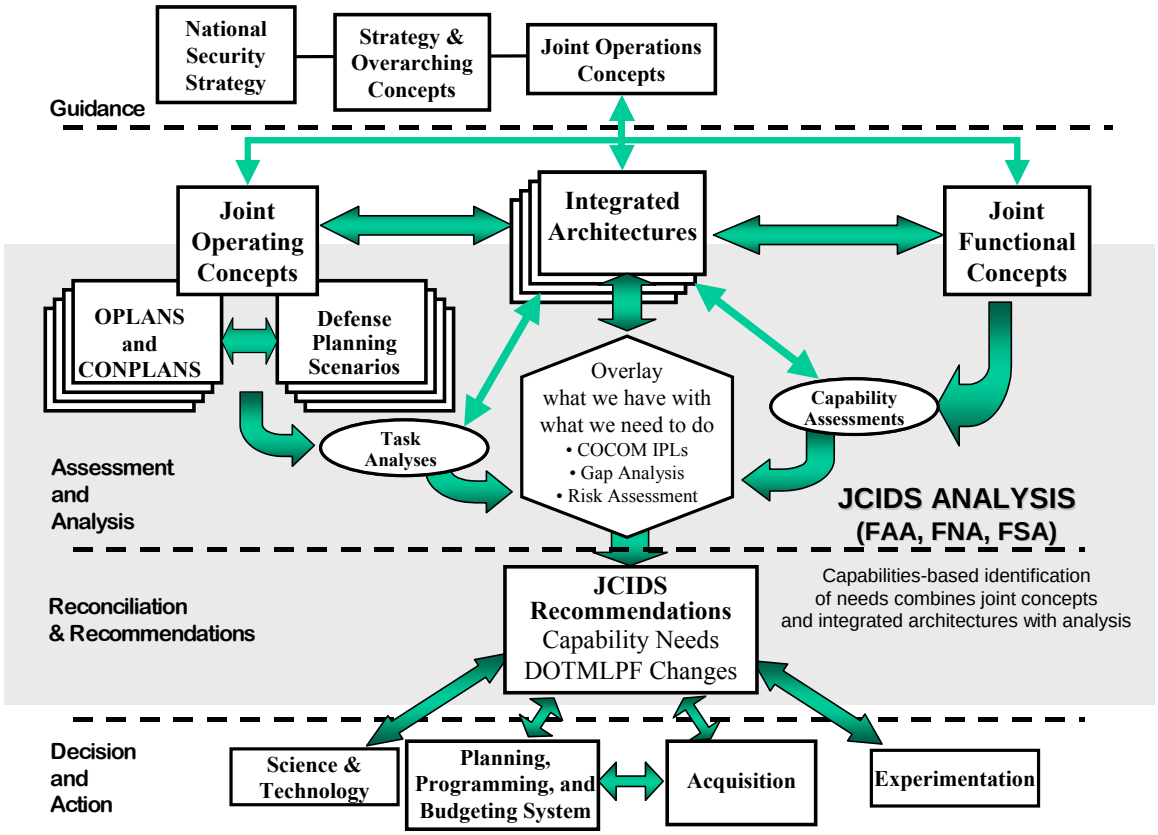


Figure 2. Top Down Capability Need Identification Process

One of the difficulties is that systems are typically scattered across different forces, programs, and engineering disciplines which are combined through the integrated architecture into a SoS. When looking at integrated architectures, to minimize development cost, there is a desire to reuse existing capabilities when further improvement is not justified, thus many of the systems would likely remain legacy elements. If only one system must be added to complete the architecture, it is typically controlled by a single program within the military that can be designed based on existing interfaces. If, on the other hand, the architecture could use or requires multiple new systems, perhaps across a variety of forces, programs, and engineering disciplines, the architecting problem becomes more difficult.

There are several examples of integrated architectures requiring multiple new systems to be developed simultaneously. This thesis will specifically examine sea basing as an example of an architecture which requires multiple new system designs. The interactions between the various systems provide a wide variety of possible combinations without a clear indication of which combination is best. To overcome this uncertainty, this thesis will explore multilevel optimization as a method to simultaneously design and evaluate different integrated architectures in order to better define possible capabilities and desired requirements.

Thesis Overview

With an understanding of a SoS and integrated architectures, the optimization problem can be examined in more detail. This thesis proposes Multiobjective Collaborative Optimization of Systems of Systems (MOCOSS) as a methodology to evaluate and optimize large-scale multiobjective SoS, such as the sea base, made up of multiobjective systems while ensuring feasibility and compatibility. Figure 3 provides a roadmap for this thesis research.

Chapter 2 will discuss various aspects of optimization. Multiobjective optimization will be defined and methods discussed to deal with the problem of determining the "optimal" solution when there is more than one objective. Multilevel optimization, specifically Collaborative Optimization (CO) will also be introduced as a way to combine multiple system designs to achieve an improved SoS design. These optimization techniques can be specifically applied to an engineering system through Multidisciplinary System Design Optimization approaches (MSDO) which is defined.

Chapter 3 will outline how the classical CO formulation can be relaxed and extended to the multiobjective situation where each individual system is itself a tradeoff of objectives. This

new formulation makes up the MOCOSS formulation. This new approach will be compared to the classical CO framework using a simple test problem.

Chapter 4 will apply MOCOSS to a real world SoS problem involving the sea base concept. The sea base concept as well as several models for system design and performance analysis used to simulate the sea base will be described. With the use of a genetic algorithm, the various models will be optimized collaboratively. Chapter 5 will provide the results from the optimization as well as more detailed analysis regarding the results and the performance of MOCOSS. The concept of a Fuzzy Pareto Front will also be introduced. Finally, Chapter 6 will describe the conclusions from this research and potential areas for future work both with MOCOSS and the sea basing problem.

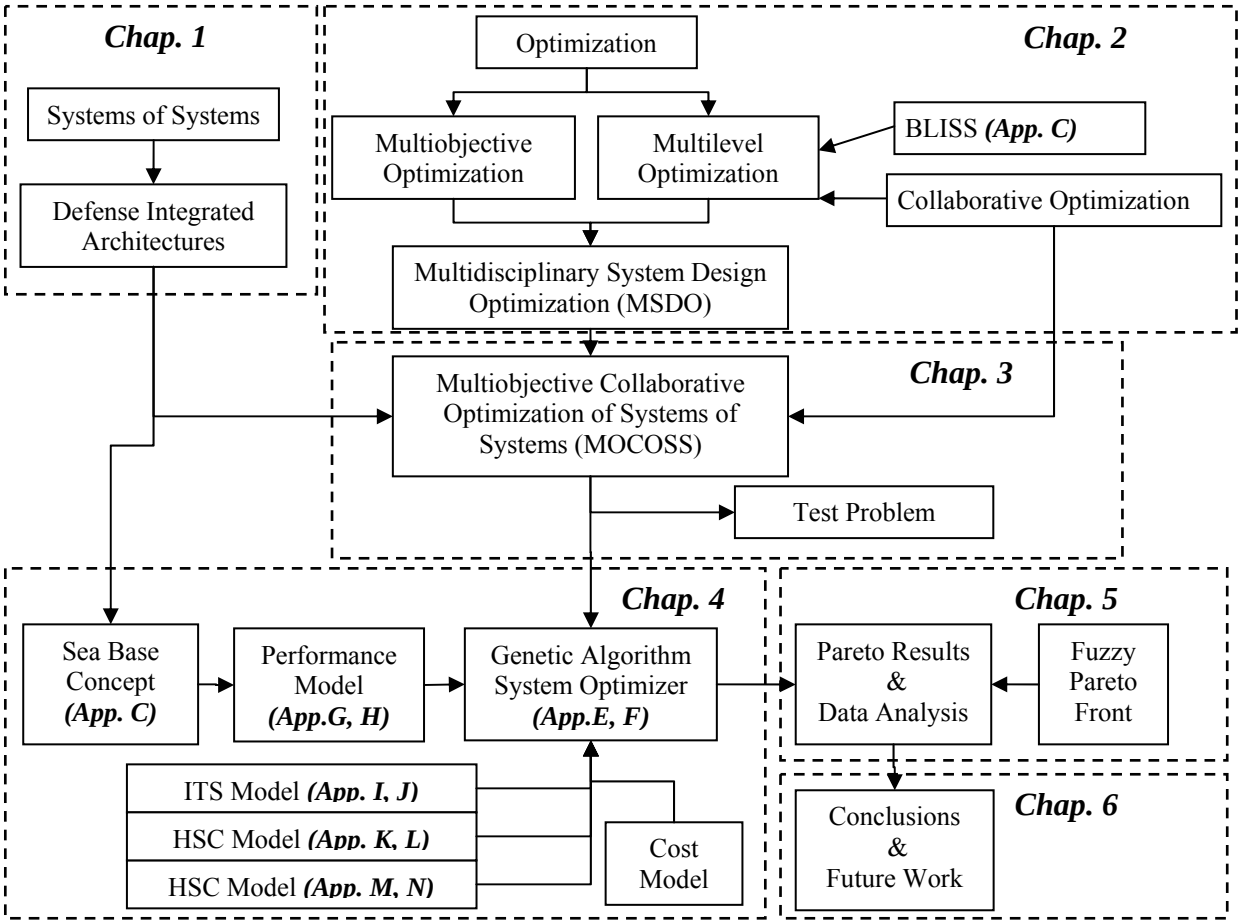


Figure 3. Thesis Roadmap

CHAPTER 2: OPTIMIZATION THEORY

Optimization

The basic optimization problem is formulated as follows:

$$\begin{aligned} &\text{minimize} && f(\mathbf{X}) \\ &\text{subject to} && \mathbf{X} \in \boldsymbol{\Omega} \end{aligned} \qquad (1)$$

where $f(\mathbf{X})$ is a real valued function called the objective function or simply J.
$\mathbf{X}$ is a vector of n real independent variables called decision variables
$\boldsymbol{\Omega}$ is the set of $\mathbf{X}$ representing the feasible set (Chong & Zak, 1996)

Equation 1 represents the general form of a constrained single objective optimization problem. While proposed as a minimization problem, it could easily be converted to a maximization problem by taking the negative of $f(\mathbf{X})$ (i.e. minimizing $-f$ is equivalent to maximizing f) without loss of generality. $\boldsymbol{\Omega}$ also usually takes the form of $\{\mathbf{X}: \mathbf{h(X)} = \mathbf{0}, \mathbf{g(X)} \leq \mathbf{0}\}$ where $\mathbf{h}$ represents the set of equality constraint functions of $\mathbf{X}$ and $\mathbf{g}$ represents the set of inequality constraint function of $\mathbf{X}$. A strict inequality [$\mathbf{g(X)} < 0$] could also be used. (Chong & Zak, 1996). Additionally, for convenience, simple boundaries on variable $\mathbf{X}$ are described by $\mathbf{X_L} \leq \mathbf{X} \leq \mathbf{X_U}$ where $\mathbf{X_L}$ and $\mathbf{X_U}$ represent the lower and upper boundaries on feasible values of $\mathbf{X}$ respectively. This leaves the more usual optimization problem form shown below.

$$\begin{aligned} &\min && J = f(\mathbf{X}) \\ &\text{s.t.} && \mathbf{h}(\mathbf{X}) = 0 \\ & && \mathbf{g}(\mathbf{X}) \leq 0 \\ &\text{D.V.} && \mathbf{X_L} \leq \mathbf{X} \leq \mathbf{X_U} \end{aligned} \qquad (2)$$

The purpose of the optimization is to find $\mathbf{X}^*$, defined as the global optimal value of $\mathbf{X}$ such that $f(\mathbf{X}) \geq f(\mathbf{X}^*)$ for all $\mathbf{X} \neq \mathbf{X}^*$. If $f(\mathbf{X})$ is multimodal (i.e. having multiple peaks) within the feasible set, there may also be local optimal values such that all values of $\mathbf{X}$ near the local $\mathbf{X}^*$ also satisfy $f(\mathbf{X}) \geq f(\mathbf{X}^*)$. Figure 4 graphically shows local and global optima.

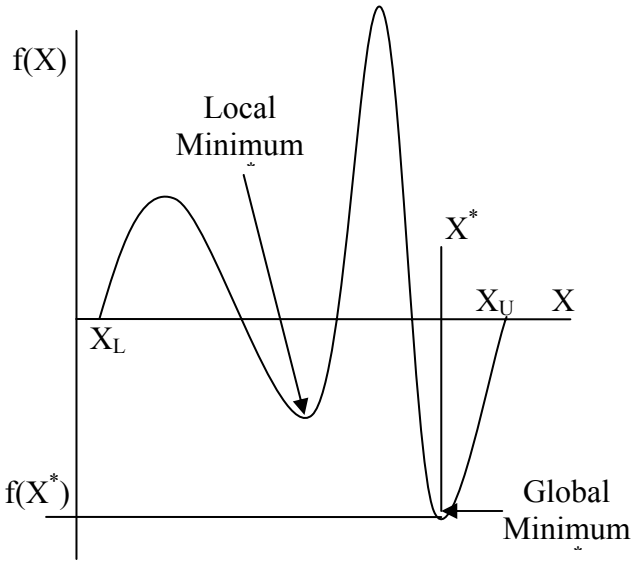


Figure 4. Multimodal Function Showing Different Minima

Depending on the type of variables, constraints, and objectives, there are several methods that can be used for optimization (e.g. Simplex for linear constrained problems, Sequential Quadratic Programming (SQP) for nonlinear constrained problems, Branch-and-Bound techniques for discrete variables, etc.). The specifics of these and other methods, their implementation, and their properties go beyond the focus of this thesis and are best left to other references (e.g. Chong & Zak, 1996; Gill, 1981).

The optimization problems discussed in this thesis will be assumed to be nonlinear and constrained. The purpose of the optimization will be design improvement. Extensions to the basic optimization framework of Equation 2 for multiobjective and multilevel optimizations will be discussed in more detail.

Multiobjective Optimization

Real engineering design decisions at the system level typically involve both technical (e.g. maximize performance) and economic (e.g. minimize cost) considerations. In general, performance and cost are opposing objectives, i.e. performance can only be increased by

increasing cost. Thus the problem becomes multiobjective. The typical multiobjective formulation is given as follows: (Liu, Yang & Whidborne, 2001).

$$\begin{array}{lll} \min & \mathbf{J} = \mathbf{f}(\mathbf{X}) = [f_1(\mathbf{X}), f_2(\mathbf{X}), ..., f_n(\mathbf{X})] & \\ \text{s.t.} & \mathbf{h}(\mathbf{X}) = 0 & \\ & \mathbf{g}(\mathbf{X}) \leq 0 & \\ \text{D.V.} & \mathbf{X_L} \leq \mathbf{X} \leq \mathbf{X_U} & (3) \end{array}$$

where **J** is a vector of n objective functions at solution **X**

With multiple objectives, generally a single solution does not exist which optimizes all objectives simultaneously, thus there are multiple "optimal" solutions. In general, the best solutions represent a compromise between the various objectives that are efficient.

Multiobjective or multicriteria problems were first examined in the field of economics where the best decision simultaneously optimizes several criteria. An economist, Vilfredo Pareto, in 1906 described the best allocation as one in which it is impossible to make one individual better off without hindering another (de Weck & Willcox, 2003). This was extended to the engineering and science fields where a solution $\mathbf{X}^*$ is considered weakly efficient if no other feasible solution exists such that $\mathbf{f}(\mathbf{X}) \leq \mathbf{f}(\mathbf{X}^*)$ and $\mathbf{f}(\mathbf{X}) \neq \mathbf{f}(\mathbf{X}^*)$ for all objectives. In other words, no other feasible solution exists which is better in one objective and at least as good in all other objectives. A strongly efficient solution $\mathbf{X}^*$ is one where no other solution exists such that $\mathbf{f}(\mathbf{X}) < \mathbf{f}(\mathbf{X}^*)$ for all objectives. All other solutions are considered inefficient (Palli, Azarm, McCluskey & Sundararajan, 1998). The efficient solutions are considered nondominated in the objective space, and represent the Pareto-optimal set, named after Vilfredo Pareto. The Pareto-optimal set also defines a curve, called the Pareto-Optimal Frontier (POF) shown graphically in Figure 5. The heavy line along the edge of the arbitrary feasible region represents the nondominated points, or the POF. Note that the POF is not necessarily a continuous curve.

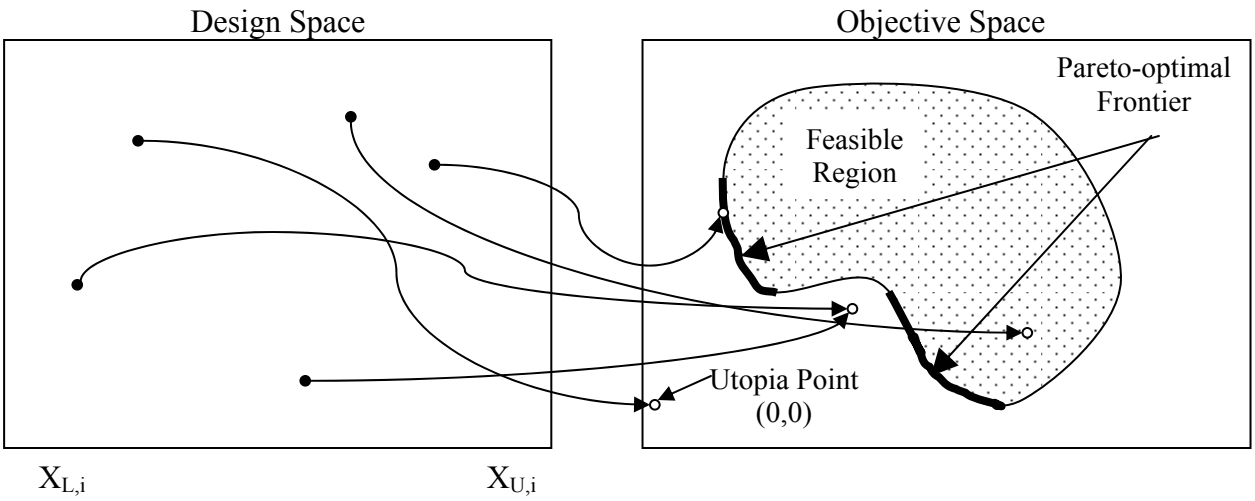


Figure 5. Pareto-Optimal Frontier of Arbitrary Design Space

Since the multiobjective problem creates multiple solutions, in order to arrive at a single preferred solution, some method must be chosen to identify the best compromise which maximizes the utility function shown in Equation (4). This equation combines the multiple objective values into a single utility, though the exact formulation of the utility function is not necessarily required. (Liu et al., 2001)

$$u[\mathbf{f(X)}] = u[f_1(\mathbf{X}), f_2(\mathbf{X}), ..., f_n(\mathbf{X})] \qquad (4)$$

The methods to obtain the best solution can be divided into four primary classes: no preference methods, a priori methods, iterative methods, and posterior methods (Palli et.al., 1998).

In no preference methods, the distance to the ideal point (defined by the combination of minimal values attainable in any objective) is minimized requiring no input from decision makers. In the a priori methods, decision makers define the utility function before any analysis through the use of priorities and/or weighting functions. Posterior methods identify the POF and then allow the decision maker to determine the best compromise solution given the range of

solutions. Finally, iterative methods identify decision maker preferences progressively as the design analysis progresses.

No preference, a priori and iterative methods essentially apply decision analysis tools in order to convert the multiobjective problem into a single objective one which can be solved using standard optimization algorithms. If we instead concern ourselves with identifying the multiobjective POF, alternative optimization methods must be used. One common approach is the use of a weighted sum shown in Equation (5). In this method, the objectives are each multiplied by a weighting factor and summed together. The optimization is then done as a single objective problem using various combinations of weights. The sum of all the weights is typically restricted to be a value of one. By varying the weights, various points along the POF can be determined. This method is easy to implement and thus frequently used.

$$\begin{aligned} & u[f(\mathbf{X})] = \lambda_1 * f_2(\mathbf{X}) + \lambda_2 * f_2(\mathbf{X}) + \ldots + \lambda_n * f_n(\mathbf{X}) \\ & \sum_{i=1}^{n} \lambda_i = 1 \\ & \lambda_i \in \Re^+ \end{aligned} \tag{5}$$

where λ_i is the weighting factor for function i

However, there are two main drawbacks. First, the weighted sum method will only identify convex regions of the POF. For example, Figure 6 shows part of an actual POF for the sea base. The concave part of the feasible region is shown between the two lines. It is in this region that a weighted sum approach will not identify Pareto solutions. Second, the solutions identified by the weighted sum method are generally not distributed evenly across the POF (de Weck & Kim, 2004).

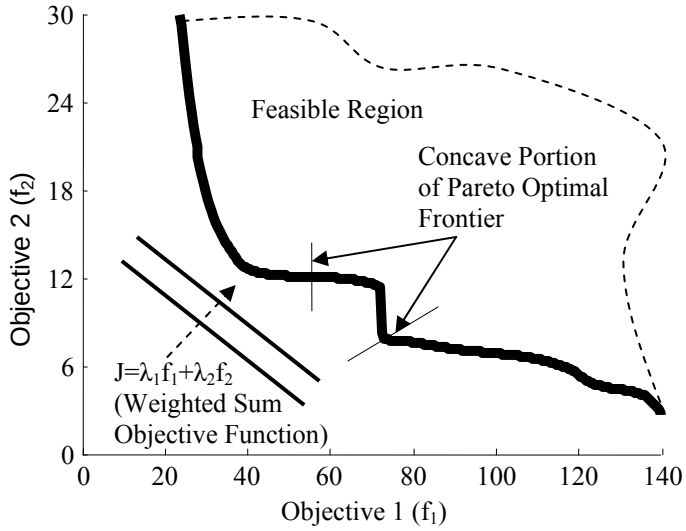


Figure 6. Actual Pareto-Optimal Frontier Showing a Concave Region and Weighted Sum Objective Function

There has been research on approaches which treat the problem as a single objective optimization while still identifying the Pareto solutions in the concave portion. These include the ε-constraint method where one objective is optimized while constraints are applied to others (Marglin, 1967), the normal boundary intersection method which identifies Pareto points by optimizing along vectors normal to a line or surface connecting the extreme points of each objective (Das & Dennis, 1998) and the adaptive weighted sum method which recursively creates subregions between identified Pareto points to identify additional Pareto-optimal designs (de Weck & Kim, 2004) to name a few.

Another approach is to use population based heuristics such as Multiobjective Genetic Algorithms (MOGA). The advantage to MOGA is that it defines a POF in a single optimization by maintaining a population of solutions (Goldberg, 1989). The optimizer examines the entire population and preferentially selects or retains through elitism the nondominated solutions of the population for subsequent generations. The end result is a set of solutions which approximates the POF. MOGA has since been applied to a wide range of difficult problems including naval ship design (Brown & Salcedo, 2003; Wolf, Dickmann & Boas, 2005).

Multilevel Optimization

Researchers working with multiobjective optimization assert that any system is actually a tradeoff of multiple objectives and should be treated as such. The same is true for multilevel optimization which recognizes that there is typically a hierarchy of decision makers with decisions made at different levels within the hierarchy (Migdalas, Pardalos & Värbrand, 1998).

The simplest approach is for each level to handle the other levels as assumptions or givens. However, this approach can result in a paradox where the assumptions drive one decision which in turn drives a decision in a completely different system, for example, the legend that the dimensions of the solid rocket boosters on the space shuttle were determined by the width of a Roman war chariot. Appendix B illustrates how one assumption leads to a design which may impose an arbitrary constraint for a subsequent design. While this whimsical example is somewhat circumstantial, there are other examples of the same decision making approach. For a Naval example, the Landing Craft, Air Cushion, (LCAC) began construction in 1981 and needed to fit within well decks of existing amphibious transport ships. At the time, those included the LPD-4 Austin and LSD-36 Anchorage classes, designed in the 1960s, which had well decks 50 feet wide. Thus the LCAC is less than 50 feet wide (Polmar, 2005; GlobalSecurity.org, 2004). Subsequently and more recently, the newly designed LPD-17 class ships had their well decks designed in order to accommodate the LCAC, thus indirectly taking on constraints established more than 30 years earlier (Pickens & Picotte, 2003). The next generation of LCAC is now being studied. It is longer than the current LCAC, "but would maintain the same height and width dimensions in order to conform to existing well deck operations" (Office of Naval Research [ONR], 2004)

In order to properly optimize the design of any system, it is important to look not only at interactions within the system but also the interactions between different systems that occur at the SoS level. The interactions, especially for complex systems, can be very difficult to assess a priori. Thus, to truly capture the impact requires concurrent design between the various parts of the hierarchy. Multilevel optimization looks at the whole of the hierarchy.

The multilevel optimization field was first introduced in 1952 by von Stackelberg who proposed a two level strategy for use in systems where policy makers at the top level influence the decisions of private individuals and companies (Migdalas et al., 1998). For example, to reduce dependence on imported energy, government could impose a tax, quotas, or rationing. Individuals and companies will adjust how they consume energy which in turn will affect pricing and import levels.

The general multilevel optimization problem is formulated as follows: (Migdalas et al., 1998):

$$
\begin{aligned}
&\min \quad f^{1}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \\
&\text{s.t.} \quad g^{1}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \le 0 \\
&\qquad h^{1}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) = 0 \\
&\text{D.V.} \quad \mathbf{X^1_L} \le \mathbf{X^1} \le \mathbf{X^1_U} \\
&\\
&\text{where } \mathbf{X^2} \text{ solves} \\
&\qquad \min \quad f^{2}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \\
&\qquad \text{s.t.} \quad g^{2}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \le 0 \\
&\qquad\qquad h^{2}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) = 0 \\
&\qquad \text{D.V.} \quad \mathbf{X^2_L} \le \mathbf{X^2} \le \mathbf{X^2_U} \\
&\\
&\text{where } \mathbf{X^k} \text{ solves} \\
&\qquad \min \quad f^{k}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \\
&\qquad \text{s.t.} \quad g^{k}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) \le 0 \\
&\qquad\qquad h^{k}(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^k}) = 0 \\
&\qquad \text{D.V.} \quad \mathbf{X^k_L} \le \mathbf{X^k} \le \mathbf{X^k_U}
\end{aligned}
\tag{6}
$$

At level 1, the highest level in the hierarchy, the decision maker has control and may only change the $\mathbf{X^1}$ variables in order to minimize the objective f^1. Level 2 can adjust the $\mathbf{X^2}$ variables

in order to minimize its objective f^2, and so on for all k levels. However, the objectives and constraints of each levels can be subject to the values of variables from other levels in which they have no control.

Multilevel optimization can also be applied to engineering systems to solve large scale coupled problems. In standard practice, the design of a single system is split among the various groups to be designed or optimized using their own internal procedures. Since the subsystems are typically not independent at the system level, there is necessary interaction between the groups. This can be handled by taking a design spiral approach where each group fixes certain areas of the design space iteratively for others until the entire system converges to a solution. Alternatively, interfaces, standards, and legacy equipment can be specified a priori, thereby decoupling the various subsystems. Each of these cases can result in suboptimizing the design and prevents full exploration of the possibilities.

Unfortunately, trying to optimize the system as a whole by coupling the systems into a single design optimization is unruly and in most cases impractical. This leads to the strategy of decomposition.

"Optimization of complicated engineering systems by decomposition is motivated by the obvious need to distribute the work over many people and computers to enable simultaneous, multidisciplinary optimization" (Sobieszczanski-Sobieski, Agte & Sandusky, 1998). Furthermore, even as computers and processes improve, it is unlikely that the optimization problem will be solved as one monolithic calculation since human judgment is still required for most problems. With complex systems, this judgment is usually best left within discipline specialists rather than at the system level. Thus, one goal is to focus on the collaboration of autonomous groups that work concurrently to compress the project elapsed time.

Decomposition focuses on optimization at lower levels with coordination at the system level. The basis for this is that most variables are at best only loosely coupled between subsystems. Thus, the system level problem can be split into smaller subsystem problems which are then coupled by a set of system level variables (i.e. Decomposition).

Decomposition performs two functions. First, it divides the problem into smaller and theoretically simpler subproblems. Second, it divides design variables for each of the subproblems into two categories, system level and subsystem level. Variables which are inputs or outputs of more than one system are considered system level variables. Variables that only affect one system are considered subsystem variables. In a well decomposed system, the number of system level variables is usually much less than the number of subsystem level variables.

While this approach is generally applied to single systems, the multilevel approach can be extended beyond the system and subsystem levels to include the SoS as well. Several multilevel methodologies for solving engineering design problems have been introduced. These include Collaborative Optimization (CO) (Braun & Kroo, 1995), Bi-Level Integrated System Synthesis (BLISS) (Sobieszczanski-Sobieski et al., 1998), and Concurrent Subspace Optimization (CSSO) (Chi & Bloebaum, 1996) to name a few. This research focused on applying CO, discussed below, combined with some elements of BLISS, discussed in Appendix C, to a SoS.

Collaborative Optimization (CO)

In CO, "a complex problem is hierarchically decomposed along disciplinary boundaries into a number of subproblems. With the use of local subsystem optimizers, each discipline is given control over its own set of local design variables and is charged with satisfying its own disciplinary constraints. The goal of each local optimizer is to agree with the other groups on values of the multidisciplinary variables, while a system level optimizer provides coordination

and minimizes the overall objective" (Braun & Kroo, 1995). This is analogous to a situation where a team leader (system level) coordinates the activities of the team members (subsystem level) through tasking in order to achieve an objective.

The system level optimizer develops a set of target system level variables. These system level variables are sent to the subsystem optimizer which has total control over subsystem variables and is also allowed to deviate from the target system level variables to achieve a feasible design. The objective of the subsystem optimizer is to minimize this departure while satisfying its own internal constraints. This optimization can be performed using any means necessary. The system level optimizer in turn adjusts the target values in such a way as to minimize the system level objective while satisfying the constraint that the subsystems do not violate the target variables. The basic structure of CO is shown in Figure 7.

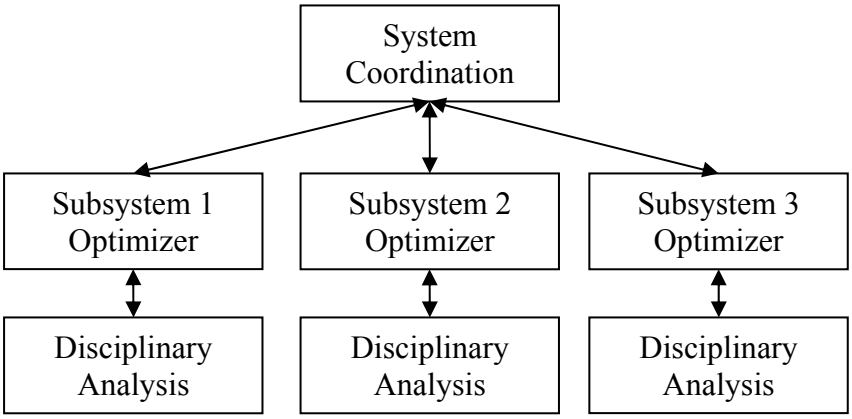


Figure 7. Block Diagram of Basic Collaborative Optimization Problem

The formulation of the subsystem objective can be any number of different functions. Many problems have used the sum of the squared differences, (e.g. $[\mathbf{Z}^{\wedge}-\mathbf{Z}^{*}]^2$) however this has shown some difficulties in convergence near the system solution (Kroo & Manning, 2000). Constraints and objectives can also be combined in a penalty method approach (DeMiguel & Murray, 2000).

This classical CO optimization can be mathematically described in Equation (7)

$$
\begin{aligned}
&\text{System Level: } \min \quad J_{sys}(\mathbf{Z^*}) \\
&\qquad\qquad\ \text{s.t.} \quad J^i = 0 \text{ for all } i \\
&\qquad\qquad\qquad\ \mathbf{Z^*_L} \le \mathbf{Z^*} \le \mathbf{Z^*_U} \\
&\\
&i^{th} \text{ subsystem: } \min \quad J^i(\mathbf{Z^*},\mathbf{Z^\wedge}) = f(\mathbf{Z^*}, \mathbf{Z^\wedge}) \\
&\qquad\qquad\ \text{s.t.} \quad \mathbf{g}(\mathbf{Z^\wedge},\mathbf{X^i}) \le 0 \\
&\qquad\qquad\qquad\ \mathbf{h}(\mathbf{Z^\wedge},\mathbf{X^i}) = 0 \\
&\qquad\qquad\qquad\ \mathbf{X^i_L} \le \mathbf{X} \le \mathbf{X^i_U} \qquad\qquad (7)
\end{aligned}
$$

where $\mathbf{Z}^*$ is a vector of system level variables defined at the system level
$\mathbf{Z}^\wedge$ is a vector of system level variables used at the subsystem level
$\mathbf{X^i}$ is a vector of subsystem i variables defined at the subsystem level

One defining difference in CO is that the system level variables are allowed to be varied by the subsystem optimizer such that the subsystem always achieves a feasible solution. This reduces the workload and communication required between levels and keeps the computational burden at the subsystem level (Braun & Kroo, 1995).

Advantages to CO are summarized as follows: (Kroo & Manning, 2000)

- Expedited integration due to minimal asynchronous communications
- Divisible computational effort
- Ability to use different subsystem optimizers
- Structure matches closely with industry and government design practices

Disadvantages to CO are summarized as follows: (Kroo & Manning, 2000)

- Greater computational time
- Requirement for careful decomposition planning and problem formulation
- Sensitivity to optimization parameters and tolerances
- Difficulty in compatibility constraints forms causing numerical difficulties
- No convergence proof (DeMiguel & Murray, 2000)

Kroo & Manning (2000) also identified areas where CO was well suited:

- Large problems requiring significant effort for subproblem solution
- Problems with low dimensionality coupling and few system variables
- Problems able to exploit special optimization methods
- Situations where tight integration is undesirable

When looking at a SoS, we can see that many of the advantages of CO apply well to this form of problem. SoS problems are large and complex because of the typically complex systems as much as the system interaction, thus CO's decomposition of optimization distributes the effort across the problem levels. SoS problems also involve diverse and minimally connected groups, both in terms of system compatibility and design team interaction. CO's minimal communication between levels and lack of communication within the subsystem level suit the SoS situation well where tight integration is typically not possible and cumbersome. The diverse groups involved in the SoS are also free to use whatever optimization technique best fits their system.

Multidisciplinary System Design Optimization

In addition to the mathematical difficulties of optimization, there are also issues involved with the construction of real world design optimization problems. Most real world designs are a combination of several disciplines such as mechanical engineering, structural engineering, electrical engineering, finance, etc. Designing a system across these different disciplines presents unique challenges. For an engineering system optimization, each discipline or subsystem typically seeks to achieve a certain objective within physical constraints. Historically, large and complex systems were developed by individual subsystems or engineering disciplines and later integrated. However, each specialized discipline would seek to optimize their subsystem within their field of expertise to create detailed designs without fully understanding the impact on the overall design in terms of performance, cost, and risk. Furthermore, different engineering fields sometimes require conflicting designs to achieve their individual objective. For instance, in a ship hull, speed performance can be improved by using a long narrow hull; however hull volume is improved by using a short and wide ship hull. These conflicting

objectives (speed versus volume) lead to conflicting designs (long and narrow versus short and wide). In order to integrate the subsystem designs across the various disciplines required either significant and costly redesign or instead of simultaneously designing the subsystems, they would be developed sequentially, where a single subsystem would be designed and then become a constraint on the next subsystem design. Both of these methods typically lead to an overall suboptimal design. Therefore, MSDO was begun as a research area for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena (Giesing & Barthelemy, 1998).

MSDO is (M)ultidisciplinary in that it seeks to integrate different discipline models together into a single macro model. MSDO deals with (S)ystems which are "a collection of things or elements which, working together, produce a result not obtainable by the things alone." MSDO focuses on engineering (D)esign. Finally, MSDO uses (O)ptimization as a mathematical framework to achieve improved performance (de Weck & Willcox, 2003).

MSDO techniques, including the multiobjective and multilevel methods discussed above, have been applied to several different complex systems such as automobiles, aircraft, telescopes, helicopters, and ships (Giesing & Barthelemy, 1998). However, each of these applications deals with a single system but could theoretically be extended to multiple systems.

CHAPTER 3: MOCOSS

Extension of CO to Multiple Objectives

The traditional CO formulation is posed as a single objective optimization problem. In most applications, the goal is to maximize the performance of the overall design by adjusting key attributes that may be common to more than one subsystem. These performance characteristics make up the system level variables. At the subsystem level, the optimizer attempts to achieve a design which does not alter the system level variables while meeting the subsystem constraints. This ensures the design is technically feasible and compatible with the other subsystem. By optimizing for the maximum system performance, the optimizer will, if it converges, arrive at a single "optimal" design.

There has been some research on multiobjective CO, (Tappeta & Renaud, 1997), which focused on the use of the weighted sum approach. This research showed different formulations and how to effectively perform sensitivity analysis. However, this research did not consider multiobjective subsystems. Some would say that each system level variable examined by the subsystem optimizer can be considered an objective, making the subsystem multiobjective; however, the CO methodology combines them into a single objective by essentially using a single special case of the weighted sum approach with each objective weight being equal.

For the SoS problem, the multiple objectives are typically performance and cost, which could themselves be multiple objectives. Within CO, the SoS would be optimized at the top or system level. The systems which make up the SoS would be optimized at the second or subsystem level. Theoretically, additional sublevels could also be included (i.e. the second or subsystem level is decomposed into various sub-subsystem levels also optimized via CO). In the case where the systems are designed independent of cost or have only one purpose, the classical

CO approach described is adequate. However, many of the individual systems which make up the SoS also have their own performance and cost objectives.

The primary issue is how the subsystem level optimizer deals with situations where there may be multiple technically feasible solutions which meet the technical system level variables. This is illustrated below in Figure 8. The isoperformance line represents the designs which achieve the desired technical performance by meeting the system level variables. Designs to the left of the POF would represent infeasible solutions, while the ones to the right are feasible. The "best" designs would be on the POF (Note: the performance objective is satisfied, while cost objective is minimized) (de Weck, Miller & Mosier, 2002).

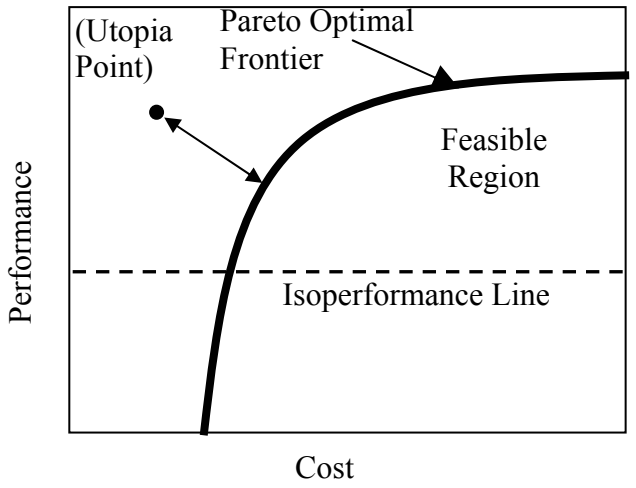


Figure 8. Depiction of Collaborative Optimization for a Multiobjective Problem

In the approach presented in previous multiobjective CO research, the system level optimizer defines a point within the objective space for each subsystem optimizer to achieve. The point could be a single pair of objectives (i.e. cost and performance levels) given to the subsystem optimizer or a group of technical and economic system level variables which are ultimately combined at the system level. The target values from the system level represent the utopia point for the subsystem level since only at that point is the subsystem objective minimized.

The system level optimizer is responsible for selecting points along the POF while the subsystems' only objective is to minimize the distance between the system level targets and a feasible solution as shown by the arrow in Figure 8. This increases the scope of the system level optimizer and requires significantly more interactions with the subsystem optimizer. It also further constrains the subsystem optimization by increasing infeasible solutions which would meet the technical system level variables but do not meet the economic system level variables.

For the SoS problem, the classical CO formulation may be relaxed by recognizing some characteristics of the problem and utilizing some aspects of BLISS (Appendix C). Of the system level variables, there are actually two classes, those that relate to performance (Z) and those that enforce compatibility (Y). The SoS interactions that deal with performance characteristics, such as speed, range, etc., of the individual systems are typically objectives of those systems. Furthermore, many of these performance characteristics can be exceeded without affecting the feasibility of the system. For instance, a ship designed for a max speed of 30 knots is still a feasible design for a ship needed to achieve a max speed of 25 knots. Recognizing this characteristic of certain performance variables allows them to be relaxed and become inequality constraints of the system design rather than essentially equality constraints as in the classical CO approach. The system level variables required to ensure the systems can interface properly (e.g. size of airplane door to interface with gangway) are system compatibility variables. These compatibility variables can be realized by either an equality constraint or by using a tolerance depending on the level of model fidelity or compatibility required.

By relaxing the requirement to exactly meet the system level variables, the design space is expanded from a single point to a region. Additionally, the subsystem optimizer no longer enforces compatibility through the objective, but rather through the constraints.

MOCOSS maintains the business analogy that created CO, extending it to include economic factors (i.e. cost). Namely, that the subsystem be responsible for achieving the desired technical performance within the compatibility constraints while minimizing cost. In this formulation, the system level optimizer would identify an isoperformance level while the subsystem optimizer seeks to satisfy that isoperformance level and then to minimize cost.

By only using solutions which minimize subsystem cost, MOCOSS makes an implicit assumption that the optimal SoS solution is made up of optimal subsystem solutions. There are three aspects which need to be addressed, compatibility variables, performance variables, and the cost objective.

The compatibility variables (Y) are the same (within a design tolerance) across the entire SoS, and thus their affect on each subsystem is included. The performance variables represent how the SoS would be operated which may be different than how the individual systems are designed. However, specifying these system level variables as performance variables with inequality feasibility constraints ensures that the subsystem can be operated at the system specified level. With a SoS performance objective based on the system level variables, the performance result may not be optimal but is operationally feasible for all the subsystems. Subsequent system level optimization can be used to improve the performance. Taking speed as an example, the system optimizer may select a target speed of 25 knots. The subsystem optimizer may possibly identify the cheapest design is one with a max speed of 30 knots perhaps because of some concavity in the subsystem POF. That design is still feasible for a speed of 25 knots. If the performance at 30 knots is better than at 25 knots, the system level optimizer should move toward that better design in subsequent iterations. If on the other hand, performance at 25 knots is better than at 30 knots, the subsystem is still able to perform at 25 knots even though it

was designed to achieve 30 knots and thus the system level optimizer will not seek a better solution (except maybe to reduce goal speed further). This creates convergence issues but not as a trade between the subsystem attempting to match the system level variables and the system level attempting to optimize those variables since altering the system level variables may not force a subsystem design change because of the use of inequality constraints.

The last restriction with MOCOSS deals with how the subsystem optimizations are combined. Since cost is the objective of the subsystem optimization, care must be taken when combining them to identify the SoS cost. First, any variables that do not monotonically increase the total subsystem cost or represent couplings between subsystems must be included within the subsystem optimization. For example, if acquisition cost is to be optimized, number of units do not have to be included since acquiring two units cost as much or more than acquiring one unit in terms of total cost. On the other hand, if operations costs are included, having two units may cost less to acquire and operate than having only one unit and thus the number of ships would need to be included in the subsystem optimization to ensure the minimal cost. Similarly, if the characteristics of one subsystem affect the cost of another system, those affects need to be included within the subsystem optimization as system level variables. With this restriction, the subsystem costs, and possibly system level variables, can be combined using any monotonically increasing function [$a + b$, $a * b$. a^b but not $a - b$, a / b, a^{-b}] since the minimum cost would consist of the minimum of each of the subsystem costs, ensuring the subsystems are on their POF.

One further note is that the acquisition cost should be based on the actual system design variables ($\hat{Z}$) determined by the subsystem optimizer since that is how the subsystem would be built while operations costs would be based on the system design variables specified by the system level optimizer (Z^*) since that is how the subsystem would be operated.

Equation (8) shows the optimization formulation for MOCOSS.

$$
\begin{array}{llll}
\text{System:} & \min & \mathbf{J_{sys}}=[f(\mathbf{Z^*}), f(J^i, \mathbf{Z^*})] & \\
& \text{s.t.} & \mathbf{g}^i \le 0 \text{ for all i} & \\
& & \mathbf{h}^i = 0 \text{ for all i} & \\
& \text{D.V.} & \mathbf{Z^*}_L \le \mathbf{Z^*} \le \mathbf{Z^*}_U & \\
& \text{D.V.} & \mathbf{Y^*}_L \le \mathbf{Y^*} \le \mathbf{Y^*}_U & \\
\\
\text{Subsystem:} & \min & J^i = f(\mathbf{X^i}, \mathbf{Z^\wedge}, \mathbf{Y^\wedge}) & \\
& \text{s.t.} & \mathbf{g}_i(\mathbf{X^i}, \mathbf{Z^*}, \mathbf{Z^\wedge}, \mathbf{Y^*}, \mathbf{Y^\wedge}) \le 0 & \\
& & f(\mathbf{Y^\wedge}, \mathbf{Y^*}) \le \varepsilon & \\
& & \mathbf{Z^\wedge} - \mathbf{Z^*} \le 0 & \\
& & \mathbf{h}_i(\mathbf{X^i}, \mathbf{Z^\wedge}, \mathbf{Y^\wedge}) = 0 & \\
& \text{D.V.} & \mathbf{X^i}_L \le \mathbf{X} \le \mathbf{X^i}_U & \\
& & \mathbf{Z^\wedge}, \mathbf{Y^\wedge} & (8)
\end{array}
$$

where $\mathbf{J_{sys}}$ is a vector of system level objectives
i varies from 1 to the number of subsystems
J^i is the i^{th} subsystem objective
$\mathbf{g}^i$ is a vector of the i^{th} subsystem inequality constraints
in addition to sub-system specific constraints, $\mathbf{g}^i$ also includes:
compatibility constraints $f(\mathbf{Y^\wedge}, \mathbf{Y^*}) \le \varepsilon$
isoperformance constraints $\mathbf{Z^\wedge} - \mathbf{Z^*} \le 0$
$\mathbf{h}^i$ is a vector of the i^{th} subsystem equality constraints
$\mathbf{Z^*}$ is a vector of system specified performance variables to be achieved by the subsystems
$\mathbf{Z^\wedge}$ is a vector of actual subsystem performance variables
$\mathbf{X_i}$ is a vector of subsystem i design variables
$\mathbf{Y^*}$ is a vector of system specified compatibility variables to be adhered to by the subsystems
$\mathbf{Y^\wedge}$ is a vector of actual subsystem compatibility variables

Test Problem

A simple test problem can be used to illustrate the computational advantage of using MOCOSS. The SoS will be made up of two systems, A and B which each have their own tradeoff between performance and cost given in Equations (9) though (12). Figure 9 shows the Pareto optimal solution of A, B, and the SoS.

$$\text{Perf}_1 \le -1./((\text{Cost}_1.^\wedge.15/15))+15 \quad (9)$$

$$\text{Perf}_2 \le -1./((\text{Cost}_2-5).^\wedge.25/20)+20 \quad (10)$$

$$\text{Perf}_{SoS}=(\text{Perf}_1+\text{Perf}_2)/2 \quad (11)$$

$$\text{Cost}_{SoS}=\text{Cost}_1+\text{Cost}_2 \quad (12)$$

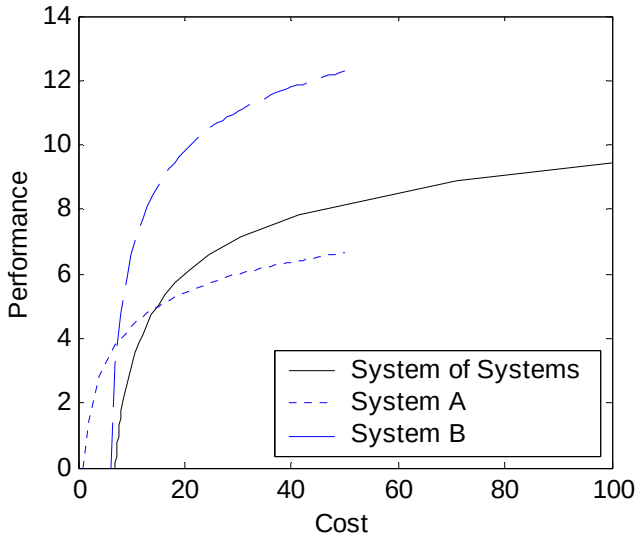


Figure 9. Actual Pareto-Optimal Frontiers for Systems and SoS Used For MOCOSS Test Problem

The test problem was solved using both the traditional CO approach (MOCO) as well as the modified approach (MOCOSS) described above. The implementation is described in Equations (13) and (14). The problem was programmed with MATLAB, with the optimization performed by a SQP optimization algorithm for both approaches at both the system and subsystem levels. The system level multiobjective problem used the weighted sum approach with seventeen different weights. Table 1 shows the results of the two simulations.

MOCO Implementation

Classical CO Implementation

System Level:	min	$\lambda * Cost_{SoS} + (1-\lambda) * Perf_{SoS}$
		$0 \leq \lambda \leq 1$
	s.t.	$J^i = 0$ for all i
	D.V.	$Perf^i*$, $Cost^i*$
i^{th} subsystem:	min	$J^i = (Perf^i* - Perf^i\hat{})^2 + (Cost^i* - Cost^i\hat{})^2$
	s.t.	Equations (9),(10)
	D.V.	$Perf^i\hat{}$, $Cost^i\hat{}$ (13)

MOCOSS Implementation

System Level: min $\lambda * Cost_{SoS} + (1-\lambda) * Perf_{SoS}$

$0 \leq \lambda \leq 1$

s.t. $\mathbf{g^i} \leq 0$ for all i

D.V. $Perf^i*$

i^{th} subsystem: min $Cost^i$

s.t. Equations (9),(10)

$Perf^i* - Perf^i\hat{} \leq 0$

D.V. $Perf^i\hat{}$, $Cost^i$ (14)

where $\mathbf{g_i}$ are the constraints of the i^{th} subsystem.

Table 1. Comparison of Test Problem Results Showing Computational Advantage of MOCOSS

	CPU Time (s)	# of System Evaluations	Average Error from System Optimum
MOCO	221.3	831	2.85e-6
MOCOSS	151.8	207	2.87e-7

Additional test using random optimization starting points and varying convergence tolerances yielded similar ratios of 1.5x less CPU Time, 3-4x fewer function evaluations, and 10x less error between MOCO and MOCOSS. The addition of compatibility constraints should reduce the relative magnitude of the difference but MOCOSS will still retain a performance advantage.

CHAPTER 4: SEA BASING MODEL

The sea base concept represents an excellent SoS problem to solve using MOCOSS. Since the concept is relatively new, first discussed in the late 1990's, the architecture is not well defined leaving a great deal of room in determining its composition. Existing platforms which might perform the various sea basing functions are nearing the end of their useful lives. This creates an opportunity for a clean sheet design of several interoperating and multidisciplinary platforms resulting in a SoS optimization problem. The types of platforms needed are also spread across different programs within the U.S. Navy creating managerial independence as well as operation independence necessary to classify it as a SoS.

The function of the sea base is relatively straight forward, dealing with the movement of materiel. This single mission allows one to overcome one of the main difficulties with SoSs, namely how to evaluate its performance. By combining the various elements of a sea base together using a mission simulation, a single performance measure can be determined. This performance measure along with cost represent a real world multiobjective, multilevel SoS problem able to be analyzed using MOCOSS.

Sea Basing Concept

President Bush stated in his 2004 State of the Union Address, "America will never seek a permission slip to defend the security of our country." Thus military commanders must be able to assure access and project power to swiftly deter or defeat enemy actions when host nation support and neighboring basing is not available (CJCS, 2004b). This issue was illustrated during the recent Iraq war where a U.S. Marine Corps division was unable to land in a NATO allied base in Turkey because the U.S. was denied permission. Sea basing seeks to address this military need.

Sea basing provides the military with the ability to rapidly deploy, assemble, equip, command, project, retrograde, and re-employ joint combat power from the sea, while providing continuous support, sustainment, and force protection to expeditionary joint forces. Supplying the necessary equipment, troops, and supplies from the sea eliminates the need for a land base or a beachhead. To reach an objective, the combat force can be taken directly from the sea base to the objective using lighters such as planes, helicopters, or air cushioned vehicles. These lighters are typically carried to the sea base and supported aboard other ships within the sea base. While pursuing the mission, the combat force must be continuously resupplied directly from the sea base. Once the mission has been accomplished, the force is reconstituted at sea to be potentially transported to another objective.

The sea base is not necessarily a single ship or platform, but is "an inherently maneuverable, scalable aggregation of distributed, networked platforms" (CJCS, 2004b, p.13). The sea base also does not operate alone. It requires sea and air dominance around the sea base, referred to as Sea Shield, which may be provided by warships and planes. Forces may also be supplied by other sea elements such as Expeditionary Strike Groups (ESG). The ships also carry additional lighters (i.e. transport vehicles able to deliver men and materiel to the objective), which could be used by the sea base to deploy its force.

The surrounding logistic system of the sea base is also important. This system consists of the joint force cargo which is made up of legacy equipment and troops. The force also requires logistical supplies such as fuel, ammunition, water, and food necessary for sustainment. For the scope of this discussion, these supplies can be assumed to have an infinite source and can be provided indefinitely to the sea base via replenishment or cargo ships. The entire sea base including the surrounding interfaces is shown in Figure 10.

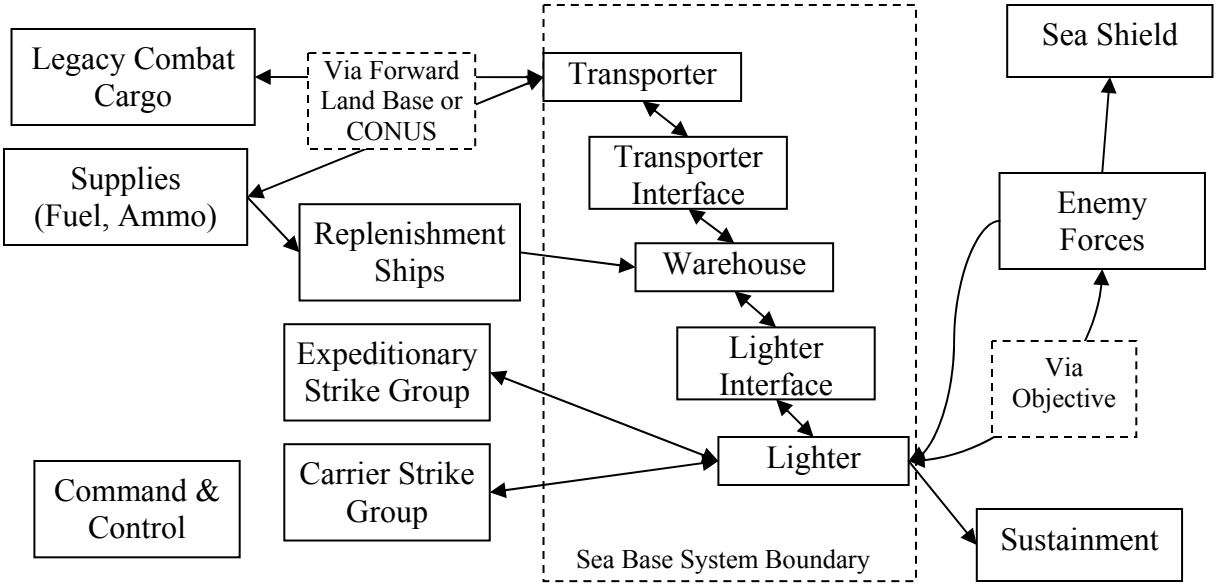


Figure 10. Sea Base Context

There are multiple ways of accomplishing the sea base concept using several different architectures. In general, accomplishing the objective from start to finish requires three types of nodes: land base, sea base, and objective nodes, illustrated in Figure 11. To travel from the land base nodes to the objective nodes, the joint force must travel aboard three platforms: a cargo platform, or transporter, which moves along the edges between the land base and sea base nodes, an assembly platform (i.e. warehouse), which makes up the sea base node, and a sea/air lift platform, or lighter, which moves the force along the edges between the sea base and the objective. In addition, there are also interfaces between the platforms. These platforms and interfaces are identified within the sea base system boundary of Figure 10. The characteristics of the platforms and interfaces, such as number, size, capacity, speed, throughput rate, etc., aggregate to determine the mission performance of the sea base.

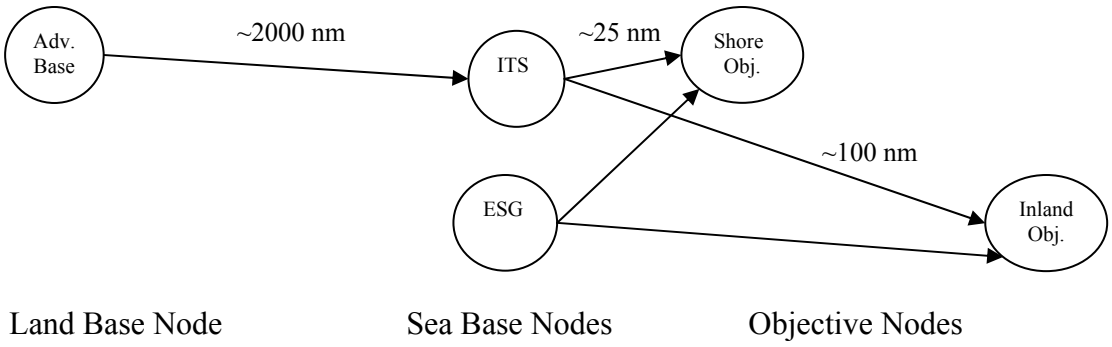


Figure 11. Nodal Diagram of Sea Base

The five blocks shown within the sea base system boundary of Figure 10 can be combined in different ways to accomplish the sea base. For instance, a potential two ship combination would have the transporter through lighter interface as one ship, and the lighter as another ship. This is similar to the Maritime Preposition Force (Future) [MPF(F)] concept where a single group of ships carries the entire cargo load to the sea base, warehouses and assembles the combat equipment, and interfaces with lighters which transport the force to the objective (Souders, Schulze, Ginburg & Goetke, 2004). Another possibility is to have a virtual warehouse such as an artificial beach (Figure 12**)** assembled at the sea base which has no cargo carrying capability and acts only as an assembly and interface point. The joint force would be delivered from the advanced base via cargo or ferry, assembled at the artificial beach**,** and then transported to the objective using lift assets from other platforms in the area. This is similar to what was done for Joint Logistics Over The Shore (JLOTS) exercises (Herron, 2001; Kane, 1998). Obviously, there is a range of possibilities between these two extremes representing the various ways the platforms could interrelate with each other.

This thesis will consider a three ship solution to achieve a sea base consisting of a High Speed Connector (HSC) to act as the cargo/ferry platform, an Intermediate Transfer Ship (ITS) to act as the sea base interface platform, and a Heavy Lift Sea Lift (HLSL) to act as the sea lift

platform which is carried and supported aboard the ITS. Figure 13 shows an operational depiction of how this sea base might operate. Appendix D is a more detailed description of the sea basing concept its function and architectural form from a system architecting perspective.

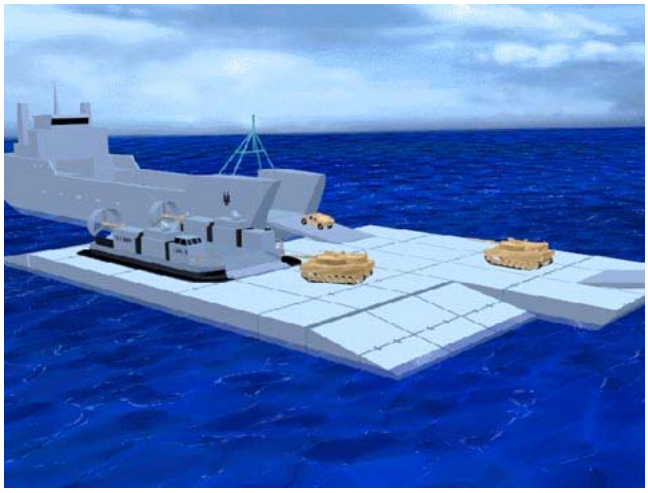


Figure 12. Artificial Beach Used in JLOTS exercise

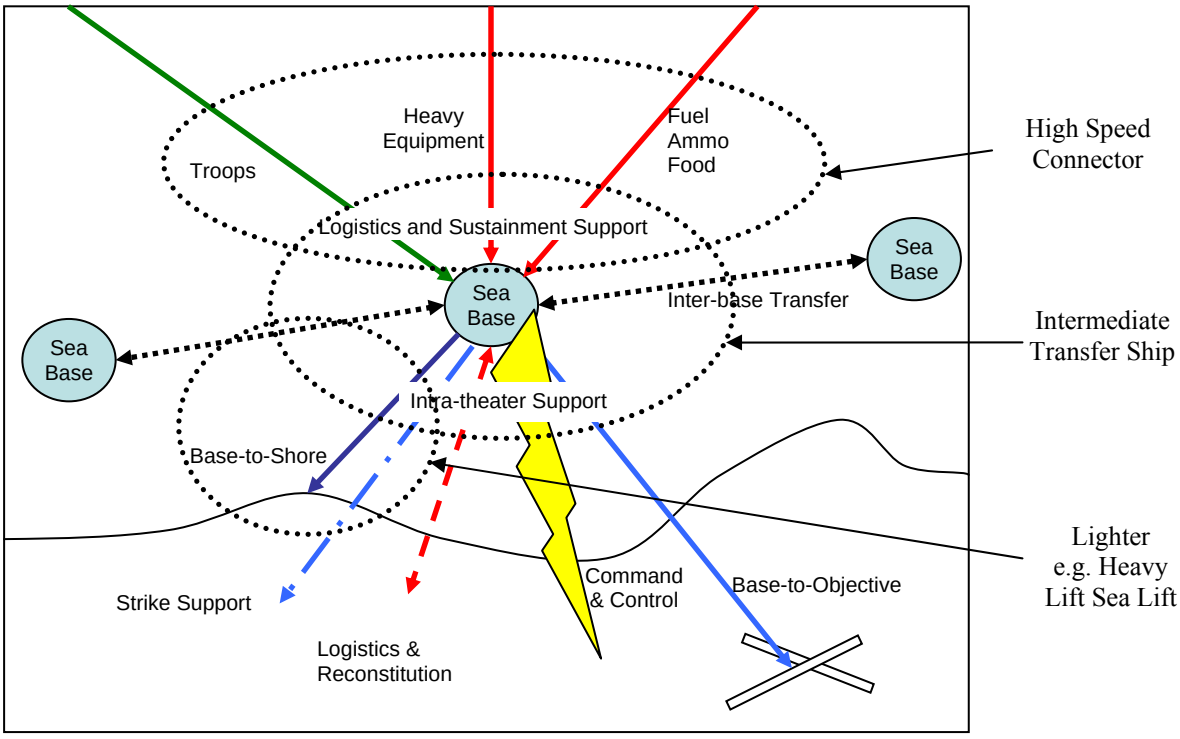


Figure 13. Sea Base Operational Diagram Showing Where Various Platforms Would Operate

Sea Base MOCOSS Formulation

The sea base problem is multiobjective in that there are two opposing goals, to minimize cost and maximizing performance. For this research, maximizing performance will be equivalent to minimizing the delivery time of the combat forces to their assigned objectives. These objectives are subject to the constraint that the sea base ship platforms are feasible. Since there are three ships to be designed (a HSC, an ITS, and a HLSL) each with their own multiple objectives, the MOCOSS approach was chosen. To clarify notation, when referring to the MOCOSS, the system level optimization is performed on the SoS while the subsystem level optimizations are performed on the ship platforms. Each subsystem is given a set of performance goals to meet. The subsystem will optimize to achieve these parameters while minimizing the individual platform cost. There is also a compatibility constraint between the ITS and HLSL to ensure they can operate together (i.e. the ITS can carry the HLSL). The total problem consists of 2 objective modules, 3 analysis models and 4 different optimizers. Of these, the objective modules, two of the analysis modules, two optimizers and the interfaces to the optimization algorithms were written specifically for this research. The remaining modules and optimizers used commercially developed algorithms. The variable and objective flow is shown in Figure 14. The two objective modules do not perform any optimization but rather combine the outputs of the subsystem optimizers and the system level variables to develop the system of system objective values, cost and delivery time. A description of each of the models and their optimizers is given below.

Sixteen system level variables (**Z***) and two compatibility constraints (**Y***) were used to combine the five models. The system variables were all scaled to enable optimization as integer values while still keeping a wide range of solutions. Table 2 identifies the variables, the models

that use them and their lower and upper bounds. Equation (15) is the generalized optimization problem to be solved. Specifics on the equations and constraints are described below and in Appendixes G, I, K, and M.

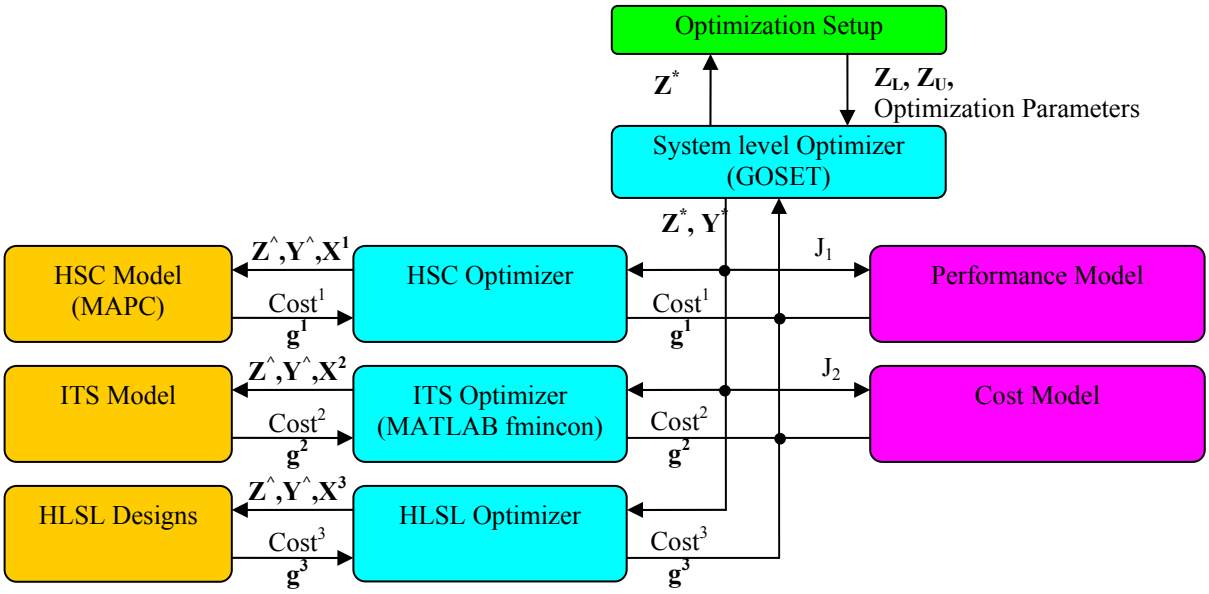


Figure 14. MOCOSS Formulation For the Sea Base Problem

Table 2. List of System Level Performance (Z*) and Compatibility (Y*) Variables

Variables	**Variable Type**	**Models Used**	**Lower Bound (Z^*_L, Y^*_L)**	**Upper Bound (Z^*_U, Y^*_U)**
HSC Speed (kts)	Z*	HSC, Perf	20	50
HSC Cargo Capacity [LT(x125), sqft(x1000)]	Z*	HSC, Perf	[2, 1]	[11, 25]
# of HSCs	Z*	Perf, Cost	1	15
ITS Cargo Capacity [sqft(x5000) Personnel(x100)]	Z*	ITS, Perf	[10, 10]	[80, 50]
ITS Interface Sites (3 types)	Z*	ITS, Perf	[1, 1, 1]	[15, 3, 2]
# of ITS	Z*	ITS, Perf, Cost	1	8
ITS Lighters (HLSL, LCAC, H-53, MV-22)	Z*	ITS, Perf, Cost	[0, 0, 0, 0]	[8, 8, 20, 20]
HLSL Speed	Z*	HLSL, Perf	40	70
HLSL Cargo Capacity (x25LT)	Z*	HLSL, Perf	3	8
HLSL Length (x10 ft)	Y*	HLSL, ITS	2	18
HLSL Length / Beam Ratio (/10)	Y*	HLSL, ITS	18	22

$$
\begin{array}{lll}
\text{System:} & \min & \mathbf{J_{sys}} = [\text{Cost}(J^i, \mathbf{Z}^*); \text{Delivery Time}(\mathbf{Z}^*)] \\
& \text{s.t.} & \mathbf{g}^i \le 0 \text{ for all i} \\
& & \mathbf{h}^i = 0 \text{ for all i} \\
& \text{D.V.} & \mathbf{Z}^*_L \le \mathbf{Z}^* \le \mathbf{Z}^*_U \\
& \text{D.V.} & \mathbf{Y}^*_L \le \mathbf{Y}^* \le \mathbf{Y}^*_U \\
\\
i^{th}\text{ Subsystem:} & \min & J^i = f(\mathbf{X^i}, \mathbf{Z}\text{^}, \mathbf{Y}\text{^}) \\
& \text{s.t.} & \mathbf{g}^i(\mathbf{X^i}, \mathbf{Z}\text{^}, \mathbf{Y}\text{^}) \le 0 \\
& & \quad 0 \le \mathbf{Y}\text{^} / \mathbf{Y}^* \le .2 \\
& & \quad \mathbf{Z}\text{^} - \mathbf{Z}^* \le 0 \\
& & \mathbf{h}^i(\mathbf{X^i}, \mathbf{Z}\text{^}, \mathbf{Y}\text{^}) = 0 \\
& \text{D.V.} & \mathbf{X^i_L} \le \mathbf{X^i} \le \mathbf{X^i}_U \\
& & \mathbf{Z}\text{^}, \mathbf{Y}\text{^} \qquad\qquad (15)
\end{array}
$$

where $\mathbf{J_{sys}}$ is a vector of system level objectives
i varies from 1 to the number of subsystems
J^i is the i^{th} subsystem objective
$\mathbf{g}^i$ is the set of i^{th} subsystem inequality constraints (including compatibility and isoperformance constraints)
$\mathbf{h}^i$ is the set of i^{th} subsystem equality constraints
$\mathbf{Z}^*$ is a vector of system specified performance variables
$\mathbf{Z}$^ is a vector of actual subsystem performance variables
$\mathbf{X}^i$ is a vector of subsystem i variables
$\mathbf{Y}^*$ is a vector of system specified compatibility variables
$\mathbf{Y}$^ is a vector of actual subsystem compatibility variables

Genetic Algorithms

A Genetic Algorithm (GA) was chosen as the system level optimizer because of its simple implementation, ability to generate a POF in a single optimization run, and its ability to handle discrete variables such as number of platforms. Other optimization methods such as SQP were attempted but proved difficult to implement.

GAs begin with a population of individuals with randomly generated genes which in this problem represent the system performance and compatibility variables. Each individual of the population is evaluated for its objective value. Feasibility constraints are accounted for as penalties to the objective to determine a fitness value. After evaluation, the population undergoes selection, where individuals are randomly selected in proportion to their fitness values (i.e. those with better fitness are selected with a higher probability). Groups of two selected

individuals, referred to as parents, undergo crossover, the equivalent to biological mating. Genes are exchanged between the two parents to create two new children to replace the parents. Finally, the population undergoes mutation where a small percentage of genes are randomly altered. These three operators, selection, crossover, and mutation create a new population. The new population is re-evaluated, completing a generation. The process of selection, crossover, mutation, and re-evaluation occurs repeatedly until the desired number of generations have passed.

The specific GA used is called GOSET and was developed at Purdue University by Dr. Scott Sudhoff. Basic genetic algorithm operators such as selection, cross-over, and mutation were used along with more advanced techniques such as elitism and diversity control. Details and explanations on the parameters used are included in Appendix E. The actual genetic algorithm interface codes are provided in Appendix F.

Performance Model

When analyzing how the sea base performs, it is important to understand the greater system dynamics of the system of ship systems. The ultimate goal of the sea base is to provide adequate logistical throughput to deploy and sustain a military force. The U.S. Marine Corps Marine Expeditionary Brigade (MEB) 2015 was chosen as the combat force for the analysis due to the availability of information on the equipment and sustainment requirements as well as the fact that the primary customer of the sea base concept is the U.S. Marine Corps. However, a different unit size or composition could also have been used. The performance model takes capability information such as cargo capacity and speed about the three ships design (HSC, ITS, HLSL) as well as a cargo load to be delivered. Additional assets such as the ESG and airlift craft are included as fixed parameters.

Using a discrete time simulation, the cargo is tracked from an advance base to one of two assigned objectives (see Figure 11). The actual distances would depend on the tactical and strategic situation in which the sea base operated. Distances would also vary with time as both the sea base and objectives would be constantly moving. For this analysis, the distances between the nodes were fixed at 2000nm between the advanced base and the sea base, 25nm between the sea base and the shore objective, and 100nm between the sea base and the inland objective. The distances were selected to be consistent with other sea base concept studies (Johnson, Wolf, West & Gold, 2005) but could have significant design impact. For instance, shortening the distance between the sea base and the shore objective would place more emphasis on the number of interfaces and time to load and unload cargo and less emphasis on speed. (U.S. Navy, 1997). Further analysis beyond this research with different distances and including stochastic position changes would be required to properly identify the impact of distance on the solution.

Cargo is treated as combat units by grouping them in sorties. The cargo is either preloaded on the ESG or ITS or remains at the advanced base for later delivery by HSCs. Cargo delivered via HSC has a time penalty aboard the ITS before it can be subsequently delivered to the objective to account for preparation and assembly time. In addition to delivering cargo, the model also accounts for the delivery of sustainment resources such as food, ammunition, and water, stored aboard the ITS, by tracking usage rates and making air drop deliveries to the objective or resupply the ITS using the HSC when supplies at the objective or ITS fall below critical levels. This sustainment requirement is included to account for the need to allocate resources to this mission during the simulation time.

The primary output of the simulation is the time to complete the delivery of the MEB combat force including any sustainment operations (i.e. resupply) required during that time.

Using combat force delivery time as the single performance measure of the sea base was used because it is quantitative, measurable, understandable, and is useful for comparison purposes. The model does provide the capability to track delivery of additional follow-on equipment and re-supply, but the time frame is much longer and less critical than the initial combat force deployment. Appendix G provides further details on the performance model and validation. The simulation code is given in Appendix H.

Cost Model

The cost model is a very simple weight based formulation which combines the number and displacements of the various subsystems using Equation (16). The k factors are used to relate the displacement in tons to a cost. Each platform has a different k factor based on the type and size. For instance, the ITS would likely have a smaller k factor than the HLSL to account for economies of scale and simpler design. The k factors used in this optimization, [4, 1, 8] for the HSC, ITS and HLSL respectively, were arbitrary and serve only to illustrate how a cost model could be implemented. More accurate formulations could be included within each subsystem model to improve the fidelity and remove the need for the k factors other than to account for volume discounts [i.e. k=f(n)]. Operations and recurring costs could also be included in the subsystem model optimization to identify life cycle costs as opposed to just acquisition costs. However, this would require including other pertinent platform characteristics within the subsystem cost model to ensure each subsystem identifies its own minimum cost.

$$\text{Cost} = \sum (k_i * N_i * \Delta_i) \qquad (16)$$

where i is the (HSC, ITS, HLSL) subsystem
k is a cost/weight factor
N is the number of platforms
Δ is the platform displacement

Platform Models

Models of each of the three platforms were developed to simulate a simple conceptual design. Each model uses a different optimization technique in order to select the best candidate platform to illustrate the flexibility of MOCOSS.

ITS Model

The ITS represents a fairly unique design combining aspects of amphibious ships with well decks and flight decks, but with cargo carrying and transfer capabilities more like a Roll-on / Roll-off (RO/RO) ship. To develop a model for the ITS, it was assumed, like most ships, to be space and volume constrained. An empirical formula from *Ship Design and Construction* for RO/RO ships, shown in Equation (17) was used an estimate of the available space.

$$\text{Length (m)} * \text{Beam (m)} * \text{Depth (m)} = 27 * \text{Lane Length (m)} \qquad (17)$$

The lane length, a typical measure of available cargo area on RO/RO vessels, was converted to total square footage. The total square footage was then allocated to various functions such as a well deck, cargo space, assembly areas, and troop accommodations with corrections made for differences in deck heights for each type of space. Additional volume for cargo tanks and ballast was also included. These required spaces were determined based on the system level variables. Further details and validation are provided in Appendix I.

In addition to available space constraints, the dimensions of the ITS were limited to what current shipyards could build as well as ensuring standard nondimensional ratios, such as length to beam, were within reasonable limits for a vessel of this type. Furthermore, a basic measure of stability, metacentric height to beam ratio, was calculated to account for intact and damaged stability requirements.

A SQP was used to vary the ship dimensions, length, beam, and depth, in order to meet the various constraints while minimizing the displacement which was a function of the dimensions. The ITS optimization interface, constraint, and objective function codes are given in Appendix J.

HSC Model

The HSC design used a model developed by the Maritime Applied Physics Corporation. It is a nonlinear empirically based model within Excel which uses existing ships for weight fractions, size ratios, and seakeeping characteristics. It is not intended to be a design tool but was used because it runs quickly, allows comparisons between various hull types and includes a basic optimization algorithm. Three hull types were examined: monohull, catamaran, and trimaran. The model allows the user to define the desired speed, payload, and range within a constrained displacement. It uses a goal programming technique where each of the goals (speed, payload, and range) are assigned a priority. The model attempts to achieve each goal in priority order by varying engines, size, and fuel load.

In the MOCOSS application, range was assumed as a given and thus given first priority. Each of the hull forms were examined for the goal speed and payload, alternating the priority between the two goals. This produced a total of six potential designs (two for each of the three hull types). An additional constraint on the cargo space was applied based on the final length and beam. The design which met all of the goals and the space constraints at the minimum displacement was selected. Additional details are included in Appendix K. The interface code is included as Appendix L.

HLSL Model

As a simplifying assumption, the HLSL was assumed to be an Air Cushioned Vehicle (ACV) similar to the LCAC currently used by the U.S. Navy. Due to the limited availability of ACV design models, existing ACV designs were used as the potential design space. In addition, some new design points were added from a design project on larger sized ACVs (Miller & Gougliodis, 2005). This gave a total of ten discrete point designs. Each of the designs included dimensions (length and beam), displacement, payload capacity, and speed and were assumed to represent nondominated designs along these variables. The optimizer selected the design that met the payload capacity and speed requirements and whose dimensions were below the actual values within a defined tolerance (20%). Of the designs meeting these criteria, the one with the minimum displacement was selected. Additional details are included in Appendix M. Implementation code is included in Appendix N.

Computational Improvement

Because GAs are population based, they tend to require more computation time relative to single solution methods. This coupled with the need to perform multiple optimizations and a time intensive simulation required efforts to reduce the evaluation time. Three techniques were used to achieve this reduction.

First, the performance simulation was skipped when any of the platforms were infeasible, and the max delivery time was assigned to the delivery time objective function. Second, the performance model was terminated after three simulated days. Since delivering the cargo in more than two days would be generally unacceptable, this change would still formally evaluate marginal solutions, but would not waste time on clearly poor solutions. The third change was to record, in a data file, the inputs and results of the optimizations. Since the optimizations and

models were all deterministic and arrive at the same result given the same input, if the inputs were repeated during a subsequent evaluation, the recorded result could be used rather then reperforming the optimization and/or simulation. Recording the inputs and results for each unique evaluation also provided a larger dataset than a single population to use in later analysis. The use of these techniques reduced the average evaluation time from 90 to 30 seconds on a Celeron 2.8GHz processor.

Page Intentionally Left Blank

CHAPTER 5: RESULTS

The Sea Basing MOCOSS optimization was done over six independent optimization runs. The first three runs, referred to as the HLSL required runs, used the basic model with the three platforms as described above and used various population and generation sizes. The HLSL required runs identified an unforeseen result of the model formulation which caused the HLSL to be adversely penalized. To check the results, the model was changed to provide the option to not include the HLSL and instead to use the existing LCAC while also accounting for the cost of obtaining additional LCACs. Three additional optimization runs, referred to as the HLSL not required runs, were done using this modification. Table 3 shows the population, generation characteristics and number of evaluations for each of the various data runs.

Table 3. Population, Generation and Evaluation Size Characteristics of the Optimization Runs

	Run #	Initial Population	Subsequent Populations	Number of Generations	# of Evaluations
HLSL Required Runs	1	50	50	50	1640
	2	500	100	50	3704
	3	100	100	100	6552
HLSL Not Required Runs	4	100	100	100	6545
	5	100	100	100	6539
	6	100	100	100	6555

Convergence Issues

In general, while GAs are known to be a generally robust optimization algorithm (Goldberg, 1989), there is important information that can be obtained from the convergence history. Figure 15 shows the generational history of the best, mean, and median values of the cost fitness function for runs 2 though 6. Run #1 was not included because the results required an adjustment due to a modeling error. Since the adjustment altered the objective function, the optimizer might have searched in another area of the design space. Therefore, the Run #1 convergence history may not be indicative of the actual optimization performance.

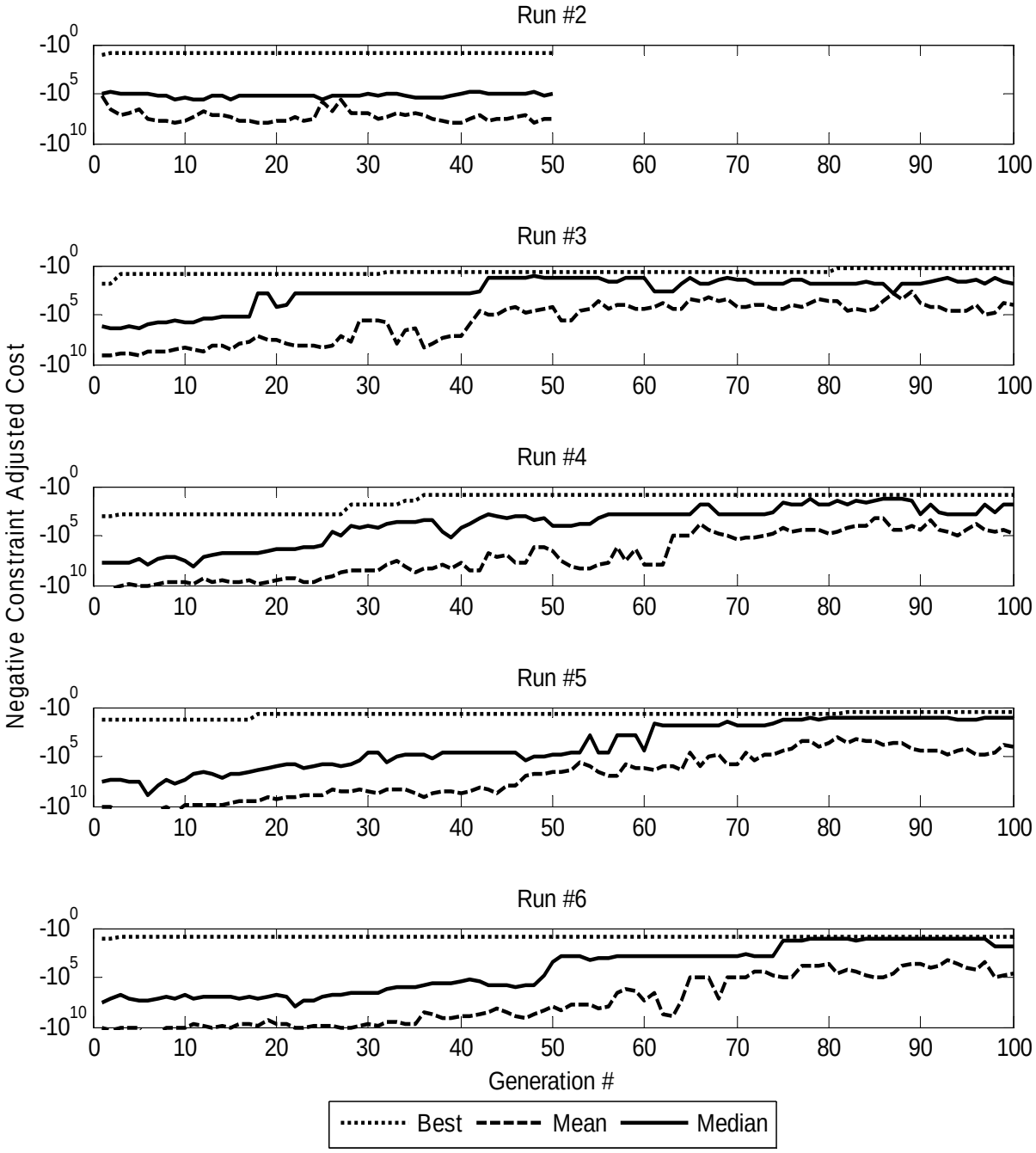


Figure 15. Cost Convergence History of Runs 2-6 Showing Improving Populations Over the Course of the Optimization

The graphs of Figure 15 plot the negative of the constraint adjusted cost fitness. The constraint adjusted delivery time fitness is not shown but exhibits similar convergence history. Cost is shown as a negative value because the desire is to minimize its value, yet the GA seeks to maximize the objective. The fitness value is shown rather than the objective value because it includes the constraint penalties, as shown in Equation (18). The constraint penalty factor of .0001 was chosen based on previous work using GOSET to provide reasonable convergence properties. This constraints penalty can alter the objective by orders of magnitude in some cases; therefore the graph is shown as a semilog plot.

$$J_f = J_o * \sum .0001 * \max(g^i(x),0) / w^i \qquad (18)$$

where J_f is the fitness value
J_o is the objective value
$g^i(x)$ is the constraint value returned by subsystem optimizer i
w^i is a scaling factor for subsystem i to bring the constraint to O(1)

Runs#3 though 6 show the GA optimization produces steady improvement over the course of the optimization in the best result, shown by the dotted line approaching one. Additionally, the mean and median values of the population also improve towards one indicating a greater portion of feasible solutions within the population. Run #2 is the exception. One cause might be speciation in which early domination prevents the GA from exploring new areas and significantly slows the population improvement. However, Figure 16 shows the distribution of the eighteen system variables, normalized between zero and one. The fact that the values are well spread would indicate good population diversity, but the designs could still have some clustering created by feasibility constraints (Wolf et al., 2005). The most likely cause is from scaling. A form of linear scaling was used in this algorithm, which works well except when most population members are fit, but a few have very low fitness (Goldberg, 1989) Since the

magnitudes of the objectives span several orders of magnitude, linear scaling can cause the majority of solutions, narrowly separated by only a few orders of magnitude, to be clustered at one end of the fitness scale while solutions which are several orders of magnitude less are at the other end. The solutions clustered at the top end will have similar chances for selection and make it more difficult to preferentially explore the very best solutions. The fact that the other runs were able to show improvement with a larger number of generations indicate that this problem could be resolved using more generations, but tuning of the GA and better scaling and penalty functions would also improve the optimization performance as well as reduce the chance of this problem occurring.

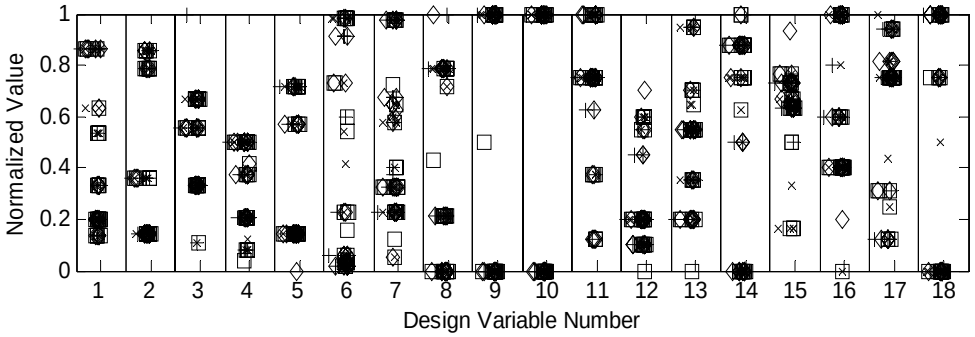
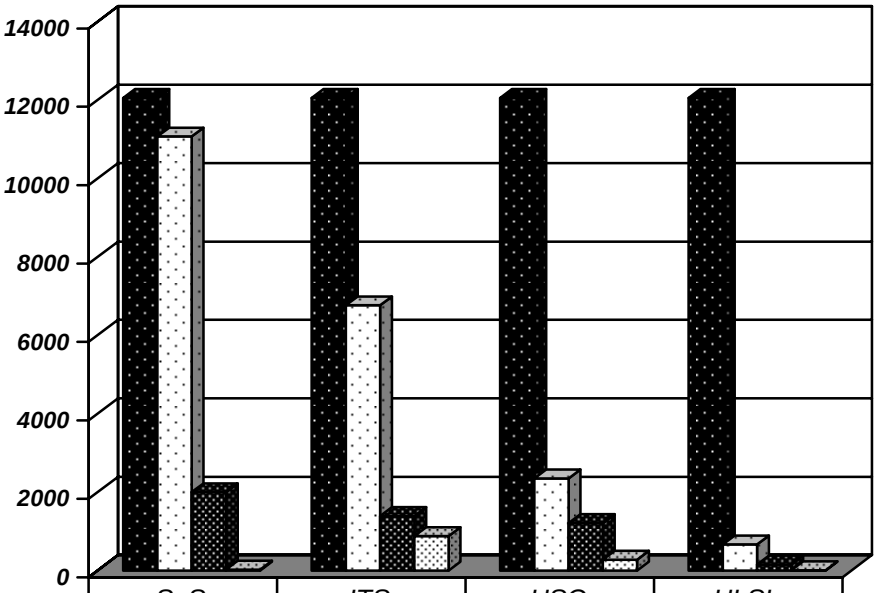


Figure 16. Gene Distribution at Generation 50 of Run #2 Showing Convergence Problem not Related to Lack of Diversity Within Population

Another convergence concern within the MOCOSS formulation is the implicit assumption that the Pareto-optimal system solutions will be made up of Pareto-optimal subsystem solutions. It is possible however, that the subsystem optimizer could identify a weakly dominated solution which may have the same displacement (cost) but lower performance characteristics. It is expected that the system level optimizer will be able to overcome this problem and move toward the Pareto solution.

To check this result, the raw data from the first three runs was split into groups based on the variables affecting each platform, or subsystem, as shown in Table 2. Duplicate and

infeasible solutions were removed from each group. To determine the Pareto solutions for each group, the system performance variables were treated as objectives minimized and maximized based on how they would affect overall performance (e.g. higher speed is better than lower speed, all else being equal). The SoS was evaluated using the final objectives (delivery time and cost), however, the number of fixed organic lighters (LCAC, MV-22, CH-53) aboard the ITS were also included to break ties when identifying Pareto-optimal solutions. This provided a penalty for using fixed platforms which were not included in the cost model. Figure 17 shows the breakdown.



	SoS	ITS	HSC	HLSL
Total	12054	12054	12054	12054
Unique	11075	6768	2350	664
Feasible	1998	1393	1209	154
Pareto	22	877	278	10

Figure 17. HLSL Required Optimization Results Showing SoS and Platform Designs Breakdown in Terms of Uniqueness, Feasibility and Dominance

To check for convergence, the 22 SoS Pareto solutions were compared to the subsystem Pareto solutions with results shown in Table 4. Most of the SoS Pareto solutions did not include

a Pareto-optimal HSC or HLSL solutions as would normally be expected. However, all the SoS Pareto solutions included an ITS Pareto solution indicating any convergence problem likely lies with the way the platform interacts with the SoS and/or with the optimizer rather than a fundamental problem with the MOCOSS method.

Table 4. Number and Percentage of SoS Solutions Containing Subsystem Pareto Solutions

ITS	HSC	HLSL
22 (100%)	10 (45%)	5 (23%)

Upon further review, the 12 non-Pareto HSC designs that were part of the SoS Pareto solutions were all dominated along the sqft capacity variable. This domination occurs because the HSC optimizer only verifies that the HSC is large enough to accommodate the desired square footage which is typically not an active constraint. Thus the displacement (i.e. cost) is only marginally affected by the desired square footage and changing this value only affects the SoS delivery time making it more difficult to drive towards the Pareto solution. In fact, the 10 SoS Pareto solutions with HSC Pareto designs consisted of only two unique HSC designs.

The number of non-Pareto HLSL solutions is also due to both the optimizer and how it interacts with the SoS. First, the HLSL model is not an optimizer per se, but merely verifies that the desired system level goals can be achieved using one of the discrete designs. It is expected that there will be differences between these discrete designs and the HLSL design being tested, however it is assumed that the desire for improved performance, will drive the system level variables toward the discrete design with no resulting increase in cost. However, the HLSL is not used in any of the SoS Pareto designs (i.e. the ITS does not carry the HLSL) and thus it is not included in the performance analysis creating no objective function change to cause the optimizer to preferentially select towards the discrete designs. Even with this difficulty at the Pareto front, the GA still identified two of the six (33%) discrete designs possible within the

system variable bounds in only 664 unique evaluations of the possible 15,810 designs (4.2%). This shows the GA performed better than random in progressing towards the discrete designs.

Data Analysis (HLSL Required)

Figure 18 shows the combined results of the HLSL required optimization runs. For comparison, the Pareto front from the first two runs were compared to the combination of all three. The third run was longer, covering 100 generations versus 50 and also increased the number of feasible SoS solutions from 258 to 1998, an almost eight fold increase. Even with more feasible solutions, the location of the POF remained relatively unchanged except for significant expansion in two regions at the extreme low and high cost ends.

One interesting area is near the labeled point. There are very few additional solutions near this point and none from run #3. This would indicate the need for additional exploration in this area.

From the first two runs, it appears that there were asymptotes along each axis. The minimum cost near 33 was attributed to having to purchase the HSC and ITS while achieving a certain level of performance. The minimum performance asymptote was attributed to the presence of the ESG which required more than one sortie to deliver its cargo. As Figure 18 shows, this performance barrier can be broken creating a concavity in the POF. By supplying sufficient organic lighters such that the sea base is able to deliver the cargo in almost a single trip, the delivery time is able to drop below 6.6 hrs. In this region, the entire surface delivered cargo is landed in one sortie, thus requiring a critical number of lighters and the ITS that carry them. Before this critical value is reached, the simulation requires a round trip back to the sea base. Another way to look at it is that there are actually two delivery times, one for the surface delivered cargo and one for the air delivered cargo. The surface delivered cargo will have a

discrete jump from 6.6 hrs to less than 1 once the critical number of lighters is reached. The air delivered cargo will also exhibit this discrete jump, but the magnitude will be much less due to the faster turn around times for aircraft sorties. The combined delivery time is the maximum of these two creating a nonlinear concavity in the possible delivery time thus justifying the use of a more advanced multiobjective optimization techniques than a simple weighted sum.

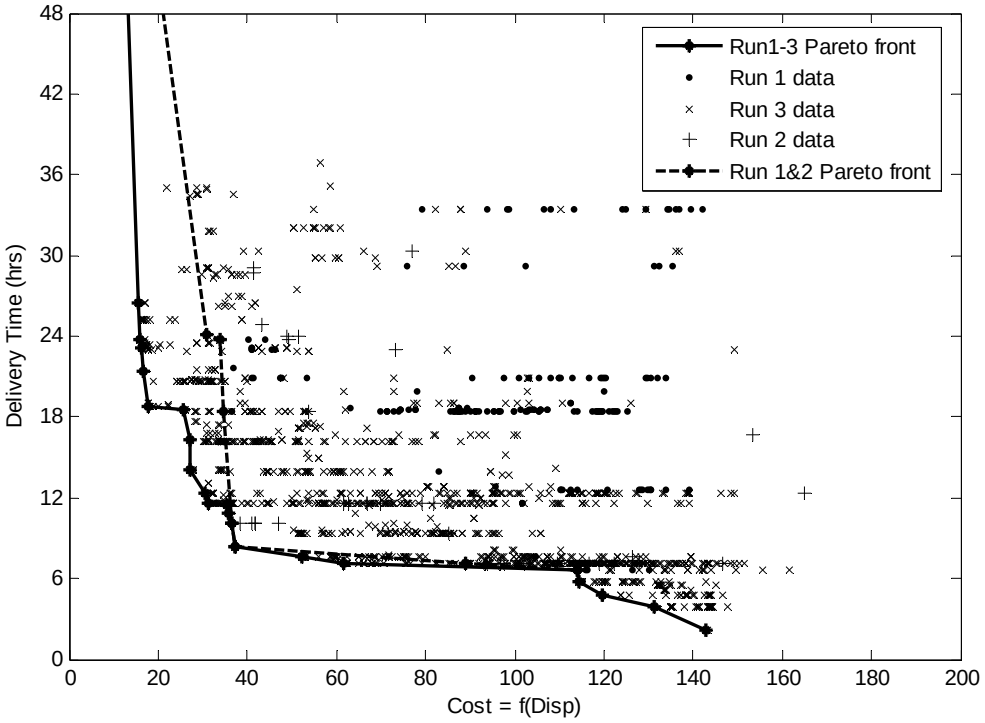


Figure 18. Plot of SoS Objective Space and Pareto-Optimal Curve for HLSL Required Optimization Results

Having the entire force for one objective land at one time does identify a problem with the model. The model does not include a limit to the number of interface sites at the objective since they are typically much less restrictive than the sea base interface sites. This model simplification therefore allows an unlimited number of lighters to offload their cargo at the objective simultaneously. This problem is only significant at the initial offload which has the most number of lighters offloading cargo. With multiple sorties between the sea base node and

the objective, bottlenecks will occurs at the sea base node minimizing the effect of any objective interface site conflict issues. However, when only one sortie is conducted, the bottleneck would instead occur at the objective. Thus the model underestimates the delivery times in the extreme low delivery time situation.

Another interesting observation from Figure 18 is that the delivery times occur at discrete intervals indicated by the lines of designs. A small contribution is the discretization of time to two minute intervals used in the performance model. The greater contribution deals with the discrete travel time (1-4 hours) for the lighters which travel between the ITS and the objective. This impact is reduced in most cases by bottlenecking at the ITS interface sites which prevents all the lighters from loading cargo simultaneously and thus separating the lighters only by the turnaround time (~15-30 min).

Table 5 looks at the 21 solutions that make up the Pareto front in Figure 18 (the 22nd point is off the graph due to a delivery time >2 days). There are some definite conclusions that can be drawn from Table 5. Within the ITS designs, there appear to be clusters of results with similar numbers and capacities. Since these clusters were developed from a single optimization (Run#3) it is possible that the cause is localized exploration rather than a preference for that particular design variant. While diversity control was enabled, the differences within the HSC and HLSL designs may have been sufficient to prevent the clustering from being adversely penalized. Additional runs with more generations might reveal more distributed results.

Table 5. System Variables of the SoS Pareto-Optimal Set For HLSL Required Optimization Runs

HSC Speed (kts)	# HSC	HSC LT (x125)	HSC sqft (x1000)	# ITS	ITS sqft (x5000)	ITS Pers (x100)	ITS flight sites	ITS well sites	ITS side sites	ITS LCAC	ITS MV-22	ITS CH-53	ITS HLSL	HLSL Speed (kts)	HLSL LT (x25)	HLSL L (x10)	HLSL L/B (/10)	Delivery Time (hrs)	"Cost" f(Disp)
21	1	3	6	8	13	14	1	1	1	7	8	16	0	40	8	17	22	2.1	142.7
21	1	3	6	8	13	14	1	1	1	7	8	12	0	40	8	17	22	4.0	131.3
22	1	3	11	8	13	14	2	1	2	7	7	7	0	46	5	17	22	4.8	119.6
22	1	3	6	8	13	14	1	1	1	7	8	6	0	46	5	17	22	5.8	114.4
40	**4**	**4**	**16**	**6**	**15**	**26**	**2**	**2**	**1**	**3**	**0**	**20**	**0**	**55**	**8**	**16**	**20**	**6.6**	**114.2**
21	1	3	7	4	13	12	5	1	2	7	8	6	0	40	8	17	22	7.1	61.6
21	1	9	11	3	20	16	6	1	2	7	0	12	0	40	8	17	19	7.6	52.3
45	**4**	**2**	**2**	**2**	**10**	**24**	**5**	**1**	**1**	**7**	**0**	**20**	**0**	**46**	**5**	**16**	**20**	**8.4**	**37.4**
24	**3**	**5**	**6**	**2**	**11**	**49**	**4**	**1**	**1**	**8**	**2**	**11**	**0**	**59**	**5**	**14**	**18**	**10.2**	**36.5**
23	1	3	12	2	13	23	6	2	1	7	13	5	0	40	5	16	21	10.9	35.7
23	1	3	12	2	13	23	6	1	2	7	11	6	0	40	8	17	19	11.5	35.5
23	1	3	12	2	13	23	6	1	2	7	5	6	0	40	5	16	22	11.6	31.2
23	1	3	12	2	13	23	6	1	2	1	4	6	0	40	5	6	22	12.4	30.5
23	1	3	12	2	13	23	6	2	1	7	0	6	0	46	5	17	21	14.1	27.2
23	1	3	12	2	13	23	6	1	2	7	0	5	0	40	5	16	22	16.3	27.0
23	1	3	12	2	13	23	6	1	2	7	0	3	0	40	4	17	22	18.5	25.6
22	1	3	11	1	20	47	6	1	1	7	0	6	0	40	5	16	22	18.7	17.6
22	1	3	6	1	20	47	6	1	1	1	0	3	0	40	4	17	22	21.4	16.6
21	1	3	6	1	20	47	5	1	1	1	0	3	0	40	4	17	22	23.1	16.3
21	1	3	6	1	20	47	5	1	1	1	0	2	0	46	5	17	21	23.8	16.0
24	1	9	11	1	13	47	5	1	2	7	0	2	0	46	5	17	22	26.5	15.5

There is also a definite trend, however, in the ITS design with a tradeoff between a large number of small capacity ITS for fast delivery times to a smaller number of large capacity ITS for low cost. Regardless of the number or size of ITS, there appears to be a preference for a large number of LCAC and a preference for CH-53 over MV-22. There is also a small tradeoff between aircraft and capacity as the small capacity ITS has more aircraft than the large capacity ITS as shown in Figure 19, though there is a great deal of scatter (R^2 of .2492). Finally, none of the Pareto solutions include a HLSL in the performance analysis.

As for the HSC designs, most of the POF includes a single slow speed HSC. The exceptions are concentrated on the solutions generated in previous runs (highlighted in bold). Since those are the minimum values for number and speed and appear across the POF, it is likely that it is cheaper and faster to put all the cargo on a set of ITS rather than ferry it via the HSC. In

fact, in performance simulations with multiple fast HSC designs, the HSC performed no deliveries and is therefore redundant. A general conclusion regarding HSCs though would be premature since there may be other missions less time critical than the initial combat force delivery that the HSC may be better suited which the model used did not account for. This possibility was not examined in this analysis.

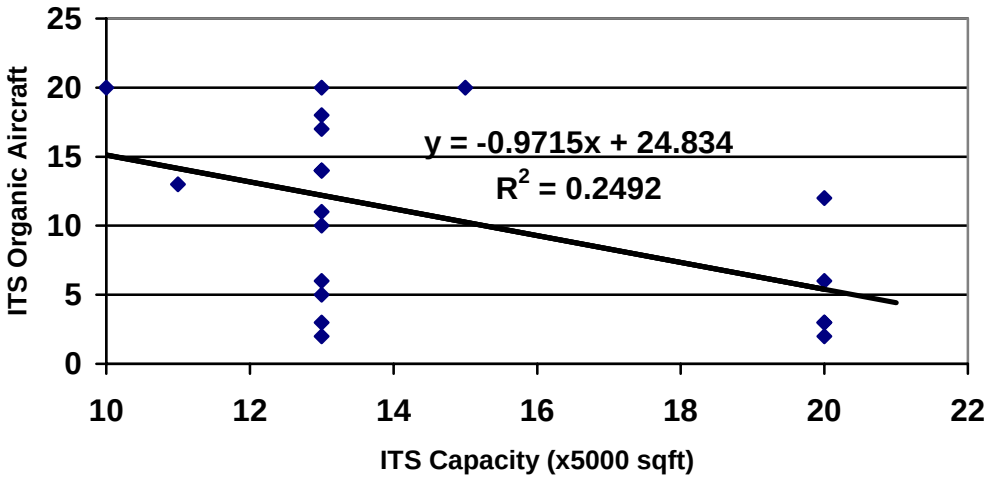


Figure 19. Plot Showing Inverse Relationship Between ITS Capacity and Organic Aircraft

Fuzzy Pareto Fronts

From Figure 18, there are several points which are very close to the Pareto front, but that are slightly dominated. However, the models used within this optimization, like any model, have potential design errors. Thus it is possible that some of these slightly dominated points might actually become dominant if more detailed design and optimization were done. Therefore, rather than excluding them when looking at Pareto-optimal results, it might be useful to look at a "Fuzzy" Pareto Front (FPF) to not only expand the number of solutions to examine but to also corroborate any conclusions. We can apply a tolerance when sorting for Pareto-optimal solutions and capture a larger data set and still retain confidence that the solutions are at least

near optimum. If we apply a 1% tolerance to the results of Runs 1, 2 and 3, we increase the dataset to 297 solutions.

The difficulty with a larger dataset is that it will include more noise and is more difficult to manage than a smaller data set such as the one in Table 5. In this formulation, some of the noise is related to the optimizer trying to eliminate redundancies. Looking at the FPF highlights these redundancies which improve the overall data and sensitivity analysis, improves the confidence in the conclusions, and highlights areas for further, more detailed design and improvement. For example, Table 6, shows a small region of the FPF with a specific delivery time with the dominant solution shown in bold. The dataset was also controlled for a specific ITS design in order to concentrate on the HSC. The nondominated solution, shown in bold, includes the low speed, single HSC design. The dominated designs show increased speed, number, and capacity which all increase the cost but have no effect on the delivery time because they are not required to deliver the combat force. This helps verify the conclusion made above that the HSC design is optimizing towards minimum performance levels and may not be required. By controlling the HSC design, similar analysis could also be performed on the ITS.

Table 6. System Variables of FPF Data Set Showing Minimal HSC Impact on Performance

HSC Speed (kts)	# HSC	HSC LT (x125)	HSC sqft (x1000)	# ITS	ITS sqft (x5000)	ITS Pers (x100)	ITS flight sites	ITS well sites	ITS side sites	ITS LCAC	ITS MV-22	ITS CH-53	ITS HLSL	HLSL Speed (kts)	HLSL LT (x25)	HLSL L (x10)	HLSL L/B (/10)	Delivery Time (hrs)	"Cost" f(Disp)
23	**1**	**3**	**7**	**4**	**13**	**12**	**12**	**1**	**2**	**7**	**8**	**6**	**0**	**40**	**3**	**17**	**19**	**7.1**	**62.4**
24	1	3	7	4	13	12	12	1	2	7	8	6	0	40	3	17	19	7.1	62.4
31	1	3	7	4	13	12	12	1	2	7	8	6	0	40	3	17	19	7.1	62.5
22	6	3	1	4	13	12	12	1	2	7	8	6	0	40	3	17	22	7.1	65.1
22	6	4	1	4	13	12	12	1	2	7	8	6	0	40	8	17	20	7.1	66.0
31	6	3	7	4	13	12	12	1	2	7	8	6	0	40	8	17	22	7.1	66.1
48	6	3	1	4	13	12	12	1	2	7	8	6	0	40	8	17	20	7.1	67.9
31	6	8	1	4	13	12	12	1	2	7	8	6	0	40	8	17	20	7.1	70.6

Data Analysis (HLSL Not Required)

One continuing trend throughout the HLSL required runs was that the HLSL was not utilized in any Pareto design. One factor was that, unlike the HLSL, LCACs could be included without increasing cost, yet provide the same function, delivering cargo from the ITS to the surface objective. Including the LCAC cost using the same cost/displacement factors as the HLSL resulted in only a small change in total cost and did not alter the POF solution set.

However, this post optimization cost adjustment potentially invalidated the optimization as designs without HLSL were preferentially selected and those with HLSL were not fully explored. To check this result, the model was adjusted to include LCAC costs. Additionally, if the HLSL was not present, a feasible HLSL design and compatibility with the ITS would not be required.

The HLSL not required runs also provide another set of independent data on the sea base problem in addition to observing how LCAC costs might affect the results. Table 7 shows the number of feasible SoS designs increased 85.2% versus only a 64.5% increase in unique SoS evaluations. This improvement can be explained by the removal of the HLSL feasibility constraint. The small change in the number of Pareto-optimal results is due to the discrete nature of the model which results in many of the delivery times overlapping those observed in the HLSL design required runs.

Table 7. Comparison of SoS Design Breakdown Showing Change in Feasibility of HLSL Not Required Optimization Results

	HLSL Not Req'd	HLSL Design Req'd	% Change
Unique	18270	11075	64.5%
Feasible	3700	1998	85.2%
Pareto	23	22	4.5%

The nondominated solutions of the HLSL not required optimization runs are given in Table 8. Once again none of the solutions include the HLSL except for ones in highest cost

region. Interestingly, the speeds and capacities are very similar to the LCAC. The HSCs were also generally slower speeds but this conclusion is not as definitive as the HLSL required runs. The same ITS trends also occur with several small ITS at the fast delivery times and few larger ITS at the low cost end.

Table 8. System Variables of the SoS Pareto-Optimal Set For HLSL Not Required Optimization Runs

HSC Speed (kts)	# HSC	HSC LT (x125)	HSC sqft (x1000)	# ITS	ITS sqft (x5000)	ITS Pers (x100)	ITS flight sites	ITS well sites	ITS side sites	ITS LCAC	ITS MV-22	ITS CH-53	ITS HLSL	HLSL Speed (kts)	HLSL LT (x25)	HLSL L (x10)	HLSL L/B (/10)	Delivery Time (hrs)	"Cost" f(Disp)
22	8	4	1	7	13	10	1	1	1	8	7	17	2	44	3	14	20	2.6	140.2
22	8	2	1	7	13	10	1	1	1	8	2	20	3	59	3	14	20	4.0	135.8
22	8	2	1	7	13	10	1	1	1	8	11	7	2	42	3	15	20	4.8	122.9
22	8	2	1	7	13	10	1	1	1	8	9	7	2	42	4	14	20	5.8	117.8
23	8	4	1	6	16	10	1	1	1	4	12	7	0	0	0	0	0	6.6	107.4
20	3	6	5	3	10	16	3	1	1	7	13	7	0	0	0	0	0	7.1	53.8
20	7	2	5	3	10	16	3	1	1	7	8	7	0	0	0	0	0	7.6	48.2
24	2	3	2	2	10	27	15	1	1	8	2	12	0	0	0	0	0	9.4	37.6
49	2	2	2	2	10	27	7	1	1	8	2	12	0	0	0	0	0	9.6	35.0
49	2	2	2	2	10	27	2	1	1	8	2	12	0	0	0	0	0	11.4	33.1
49	2	2	2	2	10	27	7	1	1	3	4	5	0	0	0	0	0	12.4	30.2
24	2	2	2	2	10	27	2	1	1	3	4	8	0	0	0	0	0	13.1	29.7
24	2	2	2	2	10	27	7	1	1	1	2	5	0	0	0	0	0	14.0	27.5
24	2	3	2	2	10	27	2	1	1	1	4	5	0	0	0	0	0	15.2	27.3
24	2	2	2	2	10	27	2	1	1	1	2	5	0	0	0	0	0	16.3	25.6
28	3	4	6	1	19	48	10	1	1	4	0	5	0	0	0	0	0	18.6	20.2
28	3	4	6	1	19	48	10	1	1	0	2	5	0	0	0	0	0	18.7	19.9
31	3	6	19	1	16	49	4	1	2	0	0	7	0	0	0	0	0	18.7	19.6
31	3	6	19	1	16	49	5	1	2	0	0	3	0	0	0	0	0	21.8	18.4
28	3	4	5	1	19	48	7	2	1	1	0	2	0	0	0	0	0	23.4	18.4
20	5	2	5	1	25	49	3	1	1	0	0	3	0	0	0	0	0	25.5	18.3

Comparison of Results

Figure 20 shows a comparison of the combined Pareto-optimal frontier curves from runs 1-3, runs 4-5, and run 6 as specified in Table 3. Each set had approximately the same number of evaluations and were performed on different computers at different times to ensure random results. The curve for runs 1-3 were adjusted to include the cost of LCAC like runs 4-6. The closeness of the three curves created from completely separate optimizations would indicate that

the Pareto-front is well defined. However, because of the random nature of the GAs, there is no guarantee that additional optimizations might not develop a curve to dominate all of these.

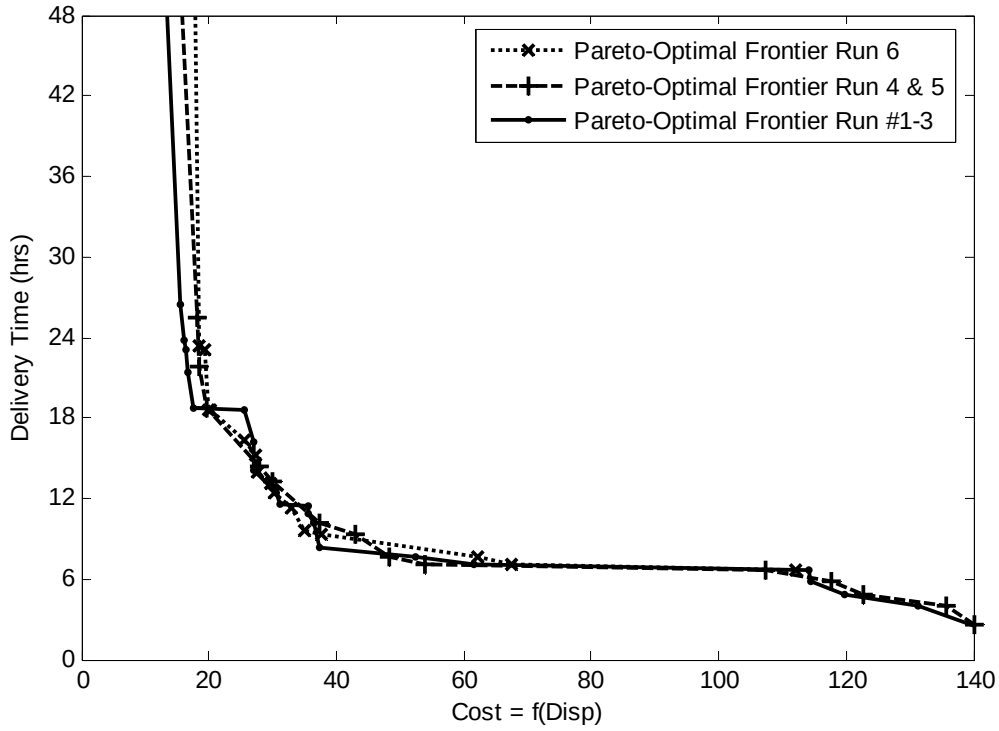


Figure 20. Comparison of Pareto-Optimal Frontiers of Different Optimizations Runs With Approximately the Same Number of Solutions

Page Intentionally Left Blank

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Conclusions

The major purpose of this thesis was to develop a formulation to optimize SoS problems where both the systems and the SoS involve a tradeoff between technical and economic factors. MOCOSS was introduced as a methodology to solve this class of problems through a relaxation of the classical CO approach. Changing the subsystem optimizer to search for the subsystem POF rather than a single point in the objective space resulted in significant improvements in both number of evaluations and accuracy of results as shown using the test problem.

Both the test problem and the sea base problem show that the MOCOSS formulation is able to arrive at a POF for a SoS which combines the subsystem POFs. The major issue as shown in the sea base problem deals with removing redundant capability which may be weakly dominated in one objective (typically performance), however this is typical of GAs which approach the POF but may not arrive at the most optimal solution unless run for a long period of time. This problem is generally not an issue in high level design where models are less accurate, but where the objective is to identify areas for more detailed exploration. For this type of problem, MOCOSS with genetic algorithms is ideal.

Another issue involved weakly dominated solutions in the subsystem optimization. The MOCOSS formulation requires detailed understanding of the problem in order to divide system level variables between those that represent goals that need to be met or exceeded from those that represent compatibility constraints between different subsystems. Relaxing the compatibility constraints using a design tolerance and allowing differences between design performance and operational performance gives greater flexibility to the optimizer. This added flexibility improves the subsystems ability to arrive at a POF solution using different model and

optimization techniques. However, when system level variables are not part of the subsystem objective function, the system level optimizer was slow to force the redundancies out of the subsystem to arrive at a system level Pareto-design which was made up of subsystem Pareto-designs as demonstrated with the HSC and HLSL models.

Computation time is another complication, even with the simplified sea base problem. The multiple subsystem POFs and large numbers of system level variables made identifying the SoS POF front very computationally intensive. The nonconvex multiobjective nature of the problem coupled with the variables being mixed-integer made GAs well suited as a system level optimizer, but this only added to the computational requirement. Several techniques such as skipping time intensive simulations for infeasible solutions and referring to previous results rather than reoptimizing a particular evaluation improved the speed by a factor of three. These techniques did not appear to negatively impact the MOCOSS formulation for the sea base. However, the computational time would make conducting detailed design analysis using MOCOSS and GAs impractical considering the thousands of design iterations required. Rather MOCOSS is best suited for use with subsystem models with response surfaces, discrete designs, or empirical relationships. Any computationally intensive processes should be kept at the system level when they can be selectively evaluated depending on the results of the subsystem optimizations. However, improved computers along with coarse grained parallel optimization processing may enable more detailed analysis.

Despite the computational inefficiencies associated with GAs, they did provide impressively consistent results. This achievement is significant considering the sea base problem had over 10^{17} possible solutions yet with about 12,000 evaluations each; three sets of independent GA optimizations describe very similar POFs. GAs also are ideally suited for

problems with discrete variables or objectives. The concave regions of the POF of the sea base also did not pose difficulties for the GA. Finally for multiobjective problems, a population of solutions enables the evaluation of FPFs to improve the understanding of the solutions space.

GAs do impose some other limitations. Potential problems with scaling and constraint application made convergence an issue. Running the optimization over more generations can typically overcome most of these issues but requires more computation time. Alternatively, the GA performance could be improved with better tuning of parameters and with a careful selection of the scaling and constraint implementation algorithm.

Another limitation of the GA is that it does not create sensitivity information like more traditional numerical optimization methods such as SQP or linear programming methods. The use of FPF was introduced in part to overcome this limitation. By evaluating the FPF data, some variables can be controlled and the sensitivity data obtained without requiring additional evaluations. The additional information provided by the FPF may increase confidence in data and trend analysis as well as identify possible redundancies that the system level optimizer failed to eliminate. Another useful aspect of the FPF is that it minimizes the effect of the inaccuracies in all models. These inaccuracies may make an actual nondominated solution appear to be dominated. Having a tolerance with the Pareto data set expands the possibilities so that the user can identify model errors which may skew results.

Finally, several conclusions were made during the data analysis of the sea base problem. It is important to realize that these conclusions are based on a model designed and written by the author and may not reflect all the required capabilities. The model was only concerned with the delivery of a specific cargo load as quickly as possible and did not include the possibility of additional cargo required for sustained operations nor for either a smaller or larger initial force

size. Other issues such as attrition, maintenance problems, varying distances, environmental conditions, etc. were also not analyzed. Also, the metric selected, delivery time, may not be the best measure of performance. Thus the purpose of this analysis was not to resolve the sea base architecture but to present how MOCOSS could be applied to SoS problem like the sea base and the results examined to obtain useful system architecture information.

Future Work

There are several areas identified during this thesis research that merit further exploration. Since the thesis explored both MOCOSS and sea basing, the possibilities for future work are separated into these two area.

MOCOSS Improvements

The mathematical rigor which went into the MOCOSS formulation needs to be improved. I asserted in this research that the requirement to decompose the objective function and delegate responsibility to the subsystem level was that the function which combined the regions must be monotonically increasing. This assertion was made based on intuitive reasoning and was submitted without proof. To be truly accepted as a formal optimization method, this assertion needs to be proven or the actual restriction made known. Specifically, it is necessary to prove under what conditions that several regions combined result in a region whose boundary is a combination of the boundaries of the other regions. The problem statement is further complicated in that the only important boundary is the nondominated section which is problem dependent.

More testing of MOCOSS also needs to be done using known test problems. The simple test problem used in this thesis did not include any compatibility constraints or subsystem variables. Additionally, only two objectives, cost and performance, were used and were

combined using a linear superposition. In general performance involves more than one objective and the subsystems may not combine in a linear fashion. Though it is not expected to significantly change the relative results, more testing may highlight other areas for improvement.

Finally, a simplification not addressed in the MOCOSS method is the performance of the individual systems outside of the SoS. By definition, these systems must be operationally independent and may operate in other SoS with different sets of systems. Perhaps this problem could be addressed by including additional objectives or somehow operating parallel MOCOSS optimizations.

Sea Base Model Improvement

As indicated before, the sea base is an excellent SoS problem which can be addressed well within the JCIDS process and use optimization techniques, specifically, MOCOSS. However, the models used in this research were written to illustrate how MOCOSS could work with different optimization methods. Furthermore, with limited time and access to computer power, it was necessary to keep the models simple. For further improvement, the models, especially the HSC, should be improved to include more subsystem level variables and to ensure, where possible, that all system level variables have an impact on cost. Additionally, it would be preferable to include cost directly within the platform models rather than the extremely simplified cost model used in this example. As for the performance model, more customer input is required to better define a performance metric which incorporates the true speed of combat force delivery, such as a time integrated cumulative combat power index, as well as the need to deliver subsequent sustained operations cargo, but not necessarily in as rapid a fashion.

The “chunking” and “smearing” effects within the performance model also need to be addressed. The “chunking effect” deals with how a cargo sortie, which represents several

vehicles, cargo, and personnel, is amalgamated into a single space and weight requirement. Each sortie can then be divided into equally size elements for delivery to the objective. The performance model includes "chucking" to maintain unit integrity, but poses problems since the division into smaller elements may not be equal. Greater fidelity in the cargo list could reduce this effect.

The "smearing effect" is where cargo is not allocated to a specific ship, but rather the total cargo on all ITS or ESG platforms is combined together and treated as a single unit. This effect ignores issues where a single cargo sortie might be spread among different ships. Likewise, interface sites are also "smeared" together. This may overlook some bottleneck issues where one ship may not have cargo, but its interface sites could effectively be used for cargo transfer from another ship. Removing the "smearing effect" would increase the complexity of the cargo distribution across the platforms and may not be worth the effort since, in practice, many of these issues are resolved with careful logistics preplanning.

There are also other architectures that could be examined besides the three platform model used in this example, such as using multiple variants of the ITS or other platforms. Additionally, some of the fixed platforms (aircraft, ESG) could also be included as designs.

Finally, within the performance model, the capability exists to include random attrition and maintenance problems. The effect of environmental factors, such as sea state, on speed and transfer capability could be included. Other factors such as varying travel distance, cargo size, beach interfaces, etc. are also important for sensitivity analysis to ensure a robust design. However, sensitivity analysis of fixed and/or stochastic system level parameters is especially difficult in this type of problem. For instance, if the cargo size was doubled, the importance of the HSC might increase as less could be carried on the ITSs. But the large amount of time

required developing even a small set of usable solutions make multiple runs more difficult to perform. One potential solution would be to limit the sensitivity and stochastic analysis to a smaller data set such as the Pareto-optimal solutions. However, limiting the analysis in this manner may not accurately reflect how these fixed and stochastic parameters affect other parts of the data set.

Page Intentionally Left Blank

REFERENCE LIST

Adams, C. (2000, February 18). *Was standard railroad gauge (4'8½") determined by Roman chariot ruts?* Retrieved April 2, 2005 from http://www.straightdope.com/columns/000218.html

Braun, R.D. & Kroo, I.M. (1995, August 14). *Development and application of the collaborative optimization architecture in a multidisciplinary design environment*. International Congress on Industrial & Applied Mathematics, Hamburg, Germany.

Brown, A. & Salcedo, J. (2003, Fall). Multiple-objective optimization in naval ship design. *Naval Engineers Journal* [electronic version]. 49-61. Retrieved April 5, 2005 from http://www.aoe.vt.edu/~brown/VTShipDesign/ASNE2002Paper.pdf

Chairman, Joint Chiefs of Staff (CJCS). (2003, June 24). *CJCS instruction on joint capabilities integration and development system*. (CJCSI 3710.01C).

Chairman, Joint Chiefs of Staff (CJCS). (2004a, July). *Joint concept development and revision plan*.

Chairman, Joint Chiefs of Staff (CJCS). (2004b, August 29). *Joint integrating concept on sea basing (Draft)*, version .25.

Chi, H.W. & Bloebaum, C.L. (1996, Sept 4-6). Concurrent subspace optimization for mixed-variable coupled engineering systems. *AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. [AIAA 96-4020].

Chong, E.P.K., & Zak. S.H. (1996). *An introduction to optimization*. New York: John Wiley & Sons Inc.

Das, I. & Dennis, J. (1998, August). Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization, 8*(3), 631-657.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, NY: John Wiley & Sons, Inc.

de Weck, O. & Kim, I.Y. (2004, April 19-22). Adaptive weighted sum method for bi-objective optimization. *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. [AIAA-2004-1680].

de Weck, O., Miller D. W., & Mosier, Gary E.. (2002, April 22-25). Multivariable Isoperformance Methodology for Precision Opto-Mechanical Systems. *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. [AIAA-2002-1420].

de Weck, O. & Willcox, K. (2003). *Multidisciplinary system design optimization*. Lecture Note, M.I.T.

DeMiguel, A.V., Murray, W. (2000, September 6-8). An analysis of collaborative optimization methods. *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* [AIAA-2000-4720].

Giesing, J.P. & Barthelemy, J.F.M. (1998, September 2-4). A summary of industry MDO applications and needs. *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* [AIAA-1998-4737].

Gill, P.E., Murray, W. & Wright, M.H. (1981). *Practical optimization*. London: Academic Press.

Gilmer, T. & Johnson, B. (1982). *Introduction to naval architecture*. Annapolis MD: Naval Institute Press.

GlobalSecurity.org. (2004, August 18). *LSD-36 anchorage class*. Retrieved April 11, 2005 from http://www.globalsecurity.org/military/systems/ship/lsd-36.htm

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization & machine learning*. Reading, MA: Addison-Wesley Publishing Company, Inc.

Herron, S.M. (2001, September-October). *JLOTS: Ship to Shore*. Retrieved April 11, 2005 from http://www.almc.army.mil/alog/issues/SepOct01/MS667.htm

Hootman, J.C. (2003). *A military effectiveness analysis and decision making framework for naval ship design and acquisition*. M.I.T. Thesis.

Johnson, A., Wolf, R., West E. & Gold, A. (2005, April 26-27). Intermediate Transfer Ship as Part of a Sea Base Family of Ships. *ASNE Day 2005.*

Kane, P. R. (1998, October). Advanced Modular Platforms for Sea State Three Operation. *1998 Naval Logistics Conference, Arlington VA, October 1998. American Society of Naval Engineers, Alexandria VA.*

Kroo, I. & Manning, V. (2000, September 6-8). Collaborative optimization: status and directions. *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. [AIAA-2000-4721].

Lamb, T. (2003). *Ship design and construction.* Jersey City, NJ: Society of Naval Architects.

Liu, G.P., Yang, J.B. & Whidborne, J.F. (2003). *Multiobjective optimisation and control*. Baldock, Hertfordshire, England: Research studies press ltd.

Maier, M. W. (1998). Architecting Principles for Systems-of-Systems. *Systems Engineering*. 1(4), 267-284.

Maier, M.W. & Rechtin, E. (2002). *The art of systems architecting*. (2nd ed.). Boca Raton, FL: CRC Press LLC.

Marglin, S. (1967). *Public investment criteria*. Cambridge, MA: MIT Press.

Migdalas A., Pardalos, P.M., & Värbrand, P. (1998). *Multilevel optimization: algorithms and applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Miller, B. & Gougliodis, G. (2005, May). *Sea base assault connector design project*. Unpublished design project. Cambridge, MA: Massachusetts Institute of Technology.

Naval Air Systems Command. (2003, April 10). *Air Capable Ship Aviation Facilities Bulletin No 1J*. (NAVAIR Lakehurst - 4.8.10.4)

Office of Naval Research. (2004, August 27). *Expeditionary logistics: Sea Base to Shore Surface Craft (SSSC).* Retrieved April 2, 2005 from http://www.onr.navy.mil/fncs/explog/explog/products/seabaseshore.asp

Palli, N., Azarm, S., McCluskey, P. & Sundararajan, R. (1998, December). An interactive multistage e-inequality constraint method for multiple objectives decision making. *Journal of Mechanical Design,* (120), 678-686.

Pickens, G.L. & Picotte, L.F.(RADM, USN Ret.) (2003, November 12). *LPD 17- A ship built by and for the expeditionary warrior*. San Antonio Class: 21st Century Amphibious Assault Ships. Retrieved April 2, 2005 from http://www.pms317.navy.mil/news/lpd17ashipbuilybyandfor.asp

Polmar, N. (2005). *The Naval Institute guide to the ships and aircraft of the U.S. fleet.* Annapolis, MD: Naval Institute Press.

Sobieszczanski-Sobieski, J., Agte, J.S. & Sandusky Jr., R.R. (1998, September 2-4). Bi-Level Integrated System Synthesis (BLISS). *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* [AIAA-1998-4916].

Sobieszczanski-Sobieski, J., Altus, T.D, Phillips, M. & Sandusky, R. (2003, October). Bilevel Integrated System Synthesis for concurrent and distributed processing. *AIAA Journal, 41*(10), 1996-2003.

Souders, R.M., Schulze, S., Ginburg, Y. & Goetke, J. (2004, April). *MPF(F) analysis of alternatives: final summary report*. Center for Naval Analysis. Alexandria, VA.

Sudhoff, S. & Lee, Y., (2004). GOSET Manual Version 1.03. (Available from Scott Sudhoff, Purdue University School of Electrical and Computer Engineering).

Tappeta, R.V. & Renaud, J.E. (1997, September). Multiobjective collaborative optimization. *Journal of Mechanical Design*, (119),403-411.

U.S. Marine Corps. (1996). *Operational Maneuver From the Sea* (Concept Paper). Quantico, VA: U.S. Marine Corps Combat Development Command.

U.S. Marine Corps. (1997). *Ship-To-Objective Maneuver* (Concept Paper). Quantico, VA: U.S. Marine Corps Combat Development Command.

U.S. Navy (1997, February). NWP 3-02.12/MCRP 3-3 1.1 A, Employment of Landing Cr2aft Air Cushion (LCAC). Norfolk, VA: Naval Doctrine Command.

Watson, D.G.M. (1998). *Practical ship design*, (1st ed.). Amsterdam: Elsevier Science Ltd.

Wolf, R., Dickmann J., & Boas, R. (2005, April 18-21). Ship design using heuristic optimization methods. *1st AIAA Multidisciplinary Design Optimization Specialist Conference*, [AIAA 2005-1980].

APPENDIXES

Page Intentionally Left Blank

Appendix A: List of Acronyms

AB	Advanced Base
ACV	Air Cushioned Vehicle
AoA	Analysis of Alternatives
BB_i	Black Boxes
BLISS	Bi-Level Integrated System Synthesis
BLT	Battalion Landing Team
CO	Collaborative Optimization
CONOPS	Concept of Operations
CSSO	Concurrent Subspace Optimization
ESG	Expeditionary Strike Groups
FoS	Families of Systems
FPF	Fuzzy Pareto Front
GA	Genetic Algorithm
GOSET	Genetic Optimization and System Engineering Tool
HLSL	Heavy Lift Sea Lift
HSC	High Speed Connector
ITS	Intermediate Transfer Ship
JCIDS	Joint Capabilities Integration and Development System
JFC	Joint Functional Concepts
JLOTS	Joint Logistics Over The Shore
JOA	Joint Operations Area
JOC	Joint Operating Concepts
JOpsC	Joint Operations Concepts
LCAC	Landing Craft, Air Cushion
LCUs	Landing Craft, Utility
LCU (R)	Landing Craft, Utility (Replacement)
LHA	Amphibious Assault Ship (General Purpose)
LHA (R)	Amphibious Assault Ship (Replacement)
LHD	Amphibious Assault Ship (Multipurpose)
LPD	Amphibious Transport Dock
LSD	Dock Landing Ship
MAPC	Maritime Applied Physics Corporation
MCCDC	Marine Corps Combat Development Command
MSDO	Multidisciplinary System Design Optimization
MEB	Marine Expeditionary Brigade
MOCO	Multiobjective Collaborative Optimization
MOCOSS	Multiobjective Collaborative Optimization of Systems of Systems
MOGO	Multiobjective Genetic Optimization
MPF(F)	Maritime Prepositioning Force (Future)
MPP	Mission Performance Parameters
MSC	Military Sealift Command
MTI	Moment to Trim 1 Inch
NATO	North Atlantic Treaty Organization
NAVSEA	Naval Sea Systems Command
OMFTS	Operational Maneuver From The Sea
OPM	Object Process Methodology
POF	Pareto-Optimal Frontier
RO/RO	Roll-on/Roll-Off
SoS	System of Systems
SQP	Sequential Quadratic Programming
SRB	Solid Rocket Boosters
STOM	Ship To Objective Maneuver
TPI	Tons Per Inch Immersion

Page Intentionally Left Blank

Appendix B: Solid Rocket Boosters and the Roman Chariot

Solid rocket boosters (SRB) used on the space shuttle are manufactured at a Thiokol plant in Utah, then shipped to Florida by rail for final assembly at the launch site. The rail line passes through one or more tunnels en route, and the SRB pieces had to be made small enough so they'd fit through the tunnel bore.

The U.S. standard railroad gauge (distance between the rails) is four feet, eight and a half inches. That's an exceedingly odd number. Why was that gauge used? Because that's the way they built them in England, and English expatriates built the U.S. railroads.

Why did the English people build them like that? Because the first rail lines were built by the same people who built the prerailroad tramways, and that's the gauge they used.

Why did 'they' use that gauge then? Because the people who built the tramways used the same jigs and tools that they used for building wagons, which used that wheel spacing.

Why did the wagons use that odd wheel spacing? Well, if they tried to use any other spacing the wagons would break on some of the old, long distance roads, because that's the spacing of the old wheel ruts.

So who built these old rutted roads? The first long distance roads in Europe were built by Imperial Rome for the benefit of its legions. The roads have been used ever since.

And the ruts? Roman war chariots made the initial ruts, which everyone else had to match for fear of destroying their wagons. Since the chariots were made for or by Imperial Rome, they were all alike in the matter of wheel spacing. Thus, the standard U.S. railroad gauge of four feet, eight and a half inches derives from the specification for an Imperial Roman army war chariot...made to be just wide enough to accommodate the back ends of two warhorses (Adams, 2000).

Page Intentionally Left Blank

Appendix C: Bi-Level Integrated System Synthesis (BLISS)

Bi-Level Integrated System Synthesis (BLISS) is a multilevel optimization technique proposed by Sobieszczanski-Sobieski, Agte, & Sandusky in 1998. BLISS relies on sensitivity analysis in order to couple the various subsystems with the system level optimization. The entire system is decomposed into separate subsystems, or Black Boxes (BB_i). The variables are divided into three classes: variables internal and unique to an individual black box, X_i, coupling variables between black boxes, $Y_{j,i}$, and system level design variables shared by at least two BB, Z. The system level objective is typically a Y variable and is calculated within a BB.

One key to BLISS is that each BB_i should optimize to the same system level objective. This ensures that the suboptimization within each BB reflects the impact of local variables on both the local and system level and results in minimization of the system objective function. To achieve this, the BLISS algorithm initializes with a best guess for Y and Z. A sensitivity analysis is performed on Y with respect to X and Z. An approximation of the system objective is made as a function of X and Z. This approximate objective function is then used as the minimization objective for each BB. Each BB performs its own optimization, varying the local variables, X_i, while treating the system variables and coupling variables from other BB as constants. The output of the BB is the objective value and other coupling variables. Sensitivity analysis is then done on the objective function of each BB as a function of Z and Y. The sensitivity analysis at each BB is used to develop an overall objective function expressed as a function of Y and Z. A system level optimization is performed to minimize the approximate objective function by varying Z. With new values for Z and Y, the process is repeated until the system converges. Figures C1 and C2 illustrate the structure and cycle of BLISS.

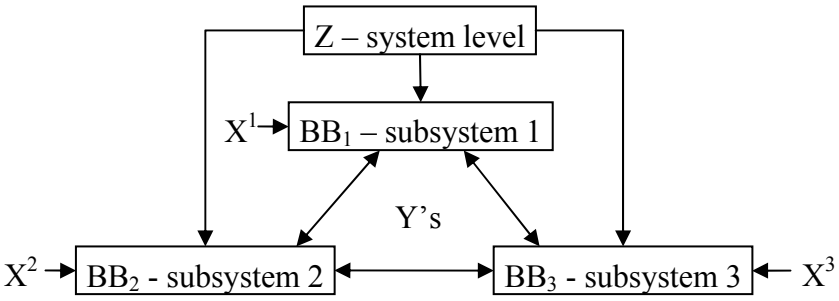


Figure C1. BLISS Variable Flow

The objective function of BLISS is presented in Equation (C1).

$$\phi^i = \sum_j D(\Phi, X_{i,j})^T \Delta X_{i,j} \qquad \text{(C1)}$$

where ϕ^i is the objective function for subsystem i

$D(\Phi, X_{i,j})$ is the total derivative of the overall objective function w.r.t. the j local variables Xi,j.

This can also be viewed as a composite objective function similar to that used in multiobjective optimization. The effect on the objective function is a sum of the local design variables weighted by their influence on the single objective of the whole system where the weights are the derivatives calculated through the sensitivity analysis.

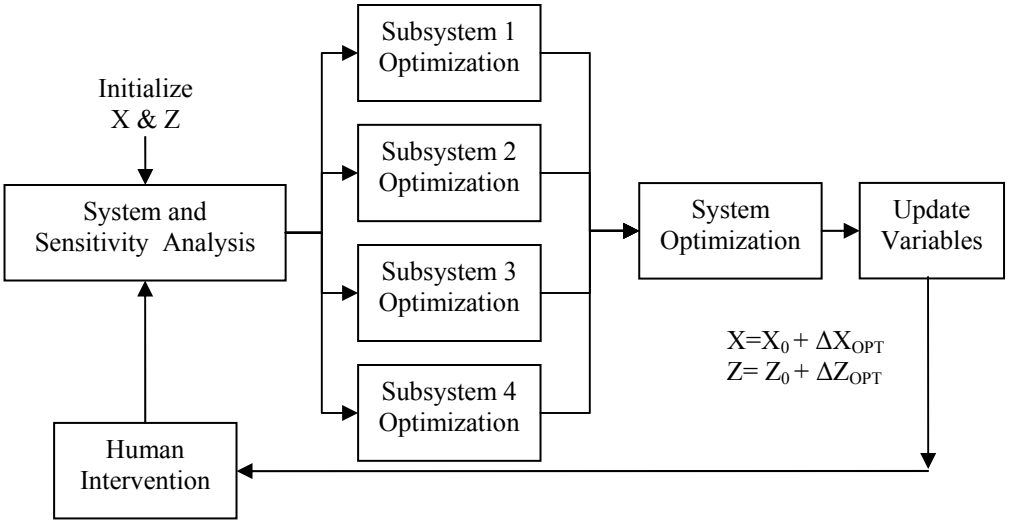


Figure C2. BLISS Cycle

One of the major drawbacks of the BLISS algorithm is the need to perform significant sensitivity analyses. Two methods were proposed to obtain the necessary derivatives. First, traditional finite differencing could be used to calculate the derivatives. This can be computationally inefficient and require several evaluations at the subsystem level. The second method is to use Lagrange multipliers as shadow prices of the various constraints and can be used to calculate the necessary derivatives. This eliminates the need for performing additional analysis, but relies on the use of an optimizer that produced Lagrange multipliers as a by product of its optimization, e.g. SQP.

BLISS 2000 (Sobieszczanski-Sobieski et al., 2003) is another method developed to minimize the sensitivity analysis by replacing the subsystem level objective function with a weighted sum of the subsystem output coupling variables Y^. This eliminates the need for calculating the total derivatives.

Appendix D: Sea Base Architecture

Introduction

Every successful military land engagement has involved some form of ground combat leading to the issue of how to get ground forces to the conflict. When conflicts erupt between neighbors, there is no concern for how to get ground forces into position or resupply them since the countries share a common border. Most of military history consists of campaigns executed in this manner where border disputes and land conquests were the primary reasons to go to war. Furthermore, the technology typically limited the force that could be brought to bear over long distances.

To reach a distant objective typically involves establishing an area of operations in a bordering country who may be an ally. If this is not possible, forces might simply pass through neutral territory to reach their objective, or may progressively take over territory to reach the ultimate objective, as Napoleon did to reach Russia. If allies cannot be found or if intimidation or seizure by force of neutral territory is unacceptable, or if natural barriers such as sea or mountain block land travel, the ground force must find another way to reach the objective

The most recent war in Iraq has shown the difficulty in utilizing allied territory to advance an attack. Turkey, a close NATO ally to the U.S. denied the 3rd U.S. Marine Corps Division from crossing its territory to attack Iraq from its northern border. This meant the force had to travel a long distance around the Arabian Peninsula in order to reach an area where it could establish a base of operations.

The U.S. must be able to address both of these concerns: Project forces over long territories and absent an allied base of operations. With two oceans separating the U.S. from its enemies, the military has spent a great deal of resources ensuring that the seas are capable of being controlled and able to be used as an avenue to reach the enemy. Furthermore, warships are considered sovereign territory and thus no permission is required in order to conduct operations from them. This naval capability has created a large amount of interest in basing operations from a sea base.

History of Sea Basing

Forces able to dominate the seas are able to use them to reach the enemy through amphibious warfare. There are examples of amphibious warfare throughout history; however, it reached its largest scale during World War II. The D-Day attack on the beaches of Normandy is a classic example of traditional amphibious warfare. Troops attacked a hostile beach where they established a beachhead. Using this beachhead, they established a port of entry to bring and store further supplies and heavier equipment in order to progress the battle further inland. This method was necessary since the sea was the only western border that the Germans had exposed to attack.

There are several problems with this method of power projection. First, D-Day and other amphibious operations of World War II required a significant amount of lead time to build up forces in a nearby base such as England. A second problem is that forcible entry against a prepared enemy is very costly in terms of casualties and time. This cost is further exasperated by the fact that the point of forcible entry is seldom the desired objective resulting in the third problem; that intermediate objectives must be seized before the mission is accomplished. In the Pacific campaign of World War II, the island hopping strategy was implemented because of the

need for nearby bases to collect forces. A fourth problem is that, as the battle progresses towards the objective, all the territory between the front line and the beachhead must be protected to prevent counterattacks. This requires a large buildup of forces before the force can advance which severely impacts the momentum and removes the element of surprise.

After World War II, the U.S. Marine Corps was assigned the primary responsibility for executing amphibious warfare. They developed various prepositioned squadrons of ships with the necessary equipment and supplies at various bases around the world. In the event of a conflict, the squadrons would immediately sail to a sea port near the conflict where the troops would join the equipment and proceed to the conflict. Additionally, the U.S. Navy developed various amphibious ships able to deploy and support a battalion sized force (~1000 troops) to execute a forcible entry on a beach. This solved the first problem of collecting forces for an assault since the force was already ready. However, this plan either required a friendly seaport or only allowed a small force to capture a beach and establish a base of operations for further buildup.

This problem was the genesis of current U.S. Marine Corps doctrine, "Operational Maneuver From The Sea (OMFTS)" and its tactical implementation, "Ship-to-Objective Maneuver (STOM)". STOM changes the traditional ship-to-shore movement to amphibious maneuver.

> Specifically, it will allow for conducting combined arms penetration and exploitation operations from over the horizon directly to objectives ashore without stopping to seize, defend, and build up beachheads or landing zones (U.S. Marine Corps, 1996).

This implies the ability to put a sufficiently sized combat ready force in a decisive position to be able to accomplish the mission. This ability to strike at 'centers of gravity' deeply inland forces the enemy to defend a larger area and increases tactical surprise (U.S. Marine Corps, 1997). This form of warfare forces the base of operations to be moved from the beachhead to a sea base

The sea base is a military concept initially developed in 1998 by the U.S. Marine Corps and included as a major component in the current naval strategy document, "Sea Power 21." It has also been declared one of five Joint Integrating Concepts by the Joint Chiefs of Staff. As defined by the Department of Defense:

> Sea basing is the ability to rapidly deploy, assemble, equip, command, project, retrograde, and re-employ joint combat power from the sea, while providing continuous support, sustainment, and force protection to expeditionary joint forces without initial reliance on land bases within the Joint Operations Area (JOA). These capabilities will enable operational maneuver and facilitate assured access and entry from the sea.

Supplying the necessary equipment, troops, and supplies from the sea eliminates the need to establish a land base or a beachhead. To reach an objective, the combat force could be taken directly from the sea base to the objective using planes, helicopters, or beachable ships such as air cushioned vehicles. While pursuing the mission, the combat force must be continuously resupplied directly from the sea. Once the mission has been accomplished, the cargo can return to the sea base or be transported to another objective.

Assuming sea control can be maintained, operations can be conducted from an invulnerable base releasing forces normally required for defense to be included in assault forces.

This increases combat effectiveness, reduces the force and logistics required, and reduces the personnel and equipment subject to enemy attack (U.S. Marine Corps, 1997).

Sea Base Context

It is important to note that the sea base is not necessarily a single ship or platform, but could be a network of different ships. The sea base also does not operate alone. It requires sea and air dominance around the sea base, referred to as Sea Shield, which may be provided by warships and planes. Forces may also be supplied by other sea elements such as carrier strike groups and ESGs. The ships in these groups typically carry additional lighters which could be used by the sea base to deploy its force. These groups could be considered a part of the sea base, but because they perform a specific aspect on a limited scope, for this discussion they will be considered only in how they support the overall goal of the sea base.

The surrounding logistic system of the sea base is also important. These logistics consist of the joint force cargo which is made up of legacy equipment and troops. The force also requires logistical supplies such as fuel, ammunition, water, and food necessary for sustainment. These supplies can be provided to the sea base via replenishment ships or aboard cargo transporters. The enemy is also critical to the context. It obviously will engage the joint force during battle which will create equipment and troop attrition which must be handled by the sea base. The enemy may also attack the sea base directly by engaging the sea shield. The enemy may also be able to directly attack lighters (platforms which move forces from the sea base to the objective) when they are at the transition of the sea shield and the combat area. Finally, command and control aspects are inherent throughout the architecture and ensure the proper execution of the joint force deployment. Figure D1 illustrates the context of the sea base and identifies the system boundary.

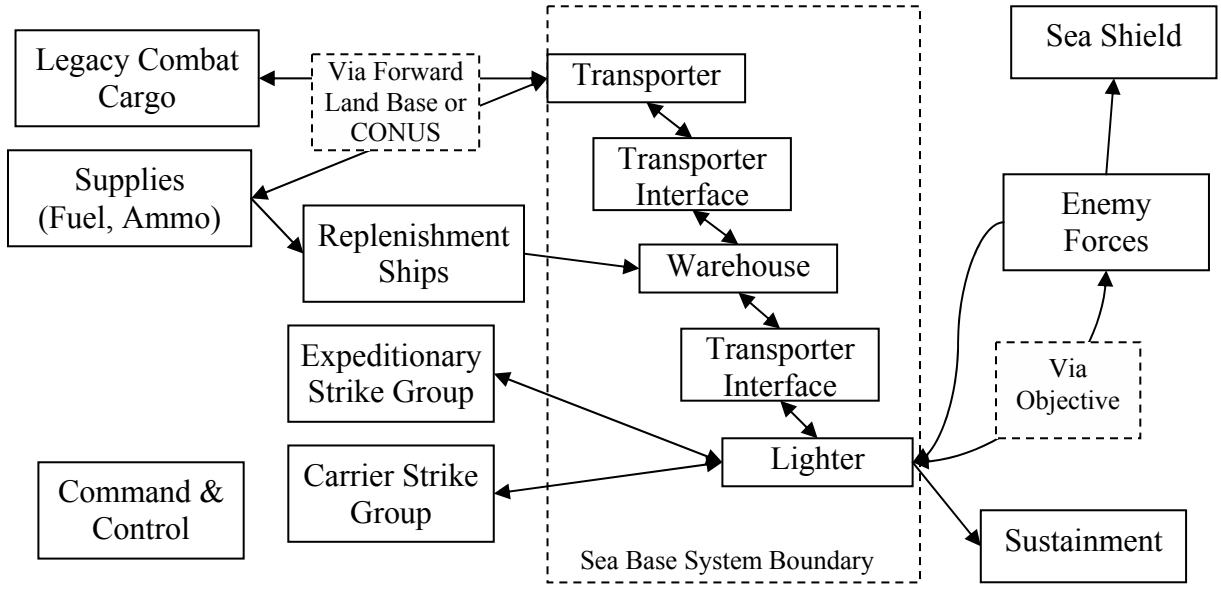


Figure D1. Sea Base Context

Sea Base Goals

The purpose of a sea base is to solve a problem consisting of several parts mentioned briefly above. First, a ground combat force must be deployed a long distance from the U.S. This force might consist of troops and/or equipment from any of the services: Army, Navy, Air Force, and/or Marines. Therefore the sea base must be joint capable and not tailored to any one force. Second, the sea base must be able to operate independent of a land base adjacent to the objective and be able to attack the objective directly. Third, the force deployed must be significant enough to achieve its mission. Since the capability exists for deploying battalion sized forces, the sea base must improve on this and be able to deploy a brigade sized force (~4000 troops) and be scalable to support larger sized forces. Finally, in order to achieve tactical surprise, the force must be capable of deployment within a single period of darkness or about 12 hours.

In summary, the problem statement is: To deploy a joint brigade sized combat force from the U.S. to a long distance objective during one period of darkness, independent of an adjacent land base. This problem statement is shown graphically in Figure D2.

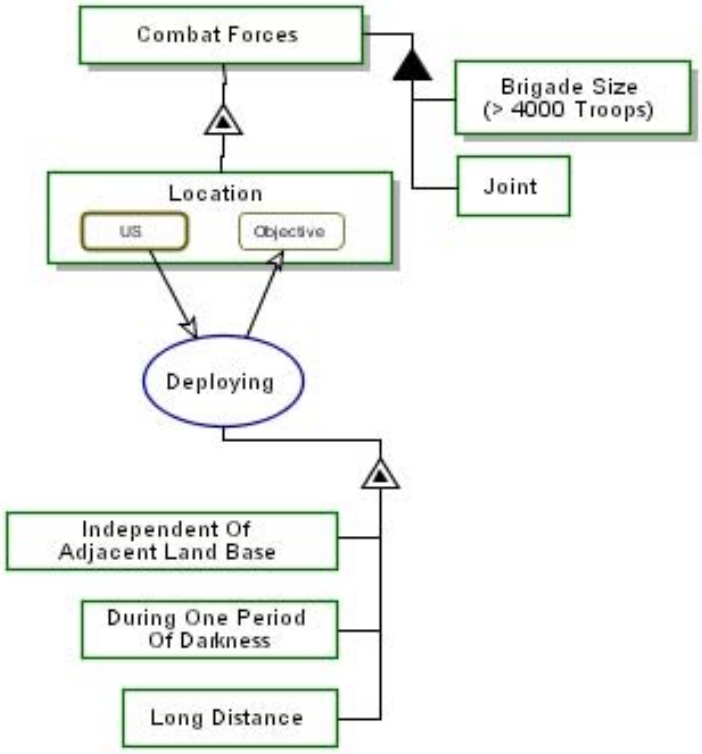


Figure D2. OPM Problem Statement Description

Conceptual Development

While the preceding section describes the sea base and makes a case for itself as a concept, there are many other possible concepts which might also satisfy the problem statement. Figure D3 shows the top level functional and conceptual development of the sea base. Three specific operations are presented which could potentially solve this problem: airlifting, selflifting, or sealifting.

Airlifting could involve a long range aircraft able to transport the force from the U.S. directly or via some intermediate airbase. Any pure airlift method, however, would be difficult to execute in the short time frame due to the small payload carrying capability of aircraft and the long travel distances.

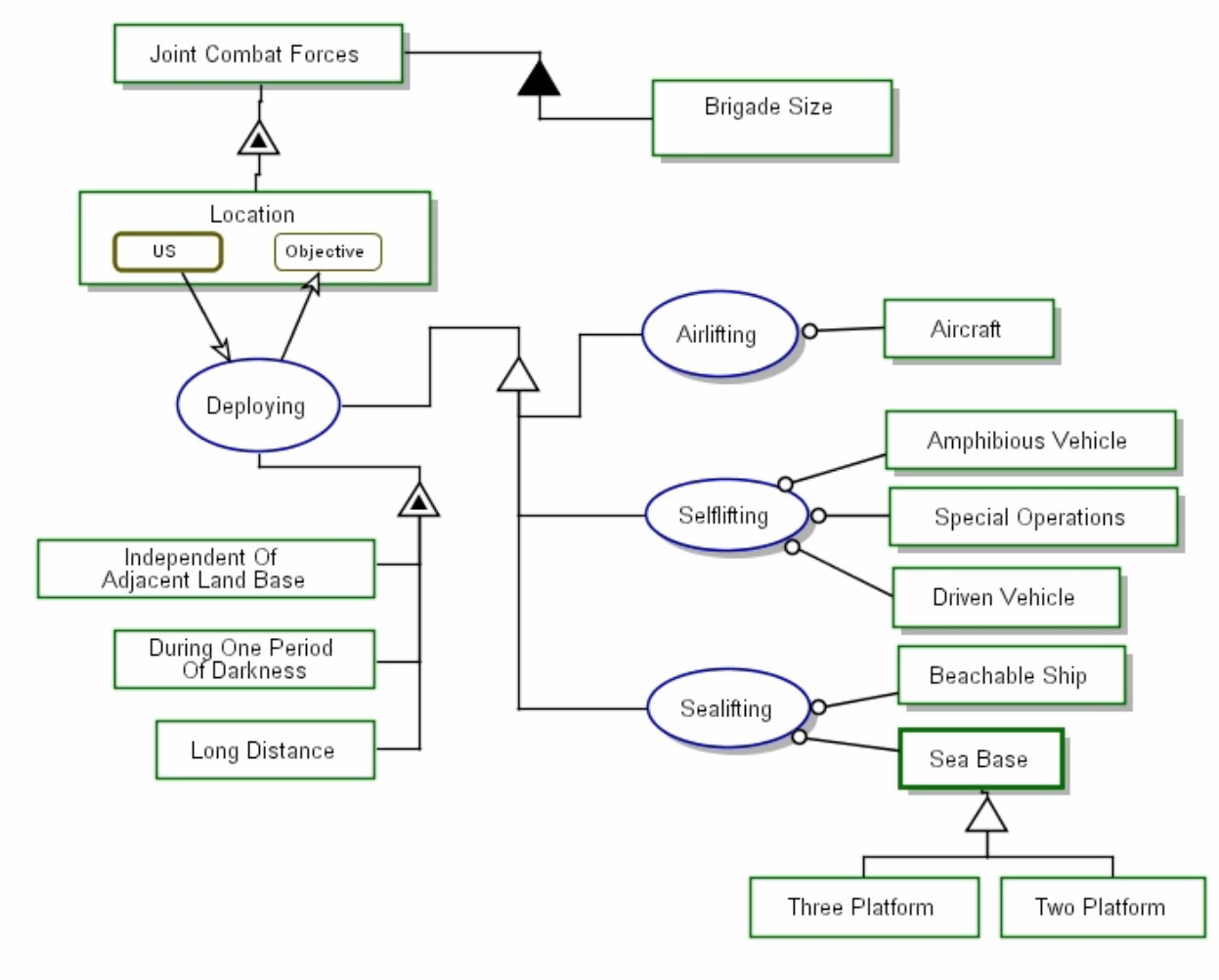


Figure D3. OPM Conceptual Development of Sea Basing

Selflifting could consist of amphibious vehicles which transport themselves over the water and then act as a combat vehicle once on shore. This concept would be technologically difficult to achieve over long distances from the U.S. to the objective and in sufficient size to meet the problem statement. This would also limit the type of equipment that could be deployed to the objective since current equipment would either be unusable or require significant modification. The combat force could drive itself to the objective for a distant land base. The problem with this concept is that it requires the permission of border countries to drive through which may not always be possible. The last selflifting concept would be to use special operations forces which have their own gear and equipment and can infiltrate and extract themselves with limited assistance. While this concept is used more frequently, it is limited in the scope and size of forces that can be deployed and would be unrealistic for a battalion or larger sized force.

A sealifting concept could be performed in one of two ways. A single ship could carry the force from the U.S. to the objective. There are ships with this capability in use today; however, the technical demands of being able to go directly to shore while also being sea worthy in the open ocean make these slow vessels. Use of these ships also assumes the objective is at or near the sea in order to attack it directly.

The final concept is to have a sea base capable of collecting the forces preparing them for deployment and then sending them to the objective. This could be performed by three types of vessels to conduct each of the phases, namely transporting forces from the U.S. to the sea base, collecting and assembling the forces in a warehouse at sea, and finally sending the forces to the objective. The sea base concept could also be accomplished using two platforms where two of the functions above are conducted by one platform such as transporting and warehousing. The three platform concept will be explored in more detail.

External Influences

There are many significant upstream and downstream influences on sea basing. On the upstream side, there are five major influences: regulation, strategy, core competencies, environment, and technology.

Regulation plays a role in the specific design of the platforms as the sea base must follow habitability standards, pollution control laws, and survivability regulations to name a few. However, given enough urgency, most, if not all of these regulations could be ignored since warships are specifically excluded from having to adhere to them.

Strategy is also critical to the architecture. At the highest generality, the need for a military or a deployed combat force is identified by national security strategy. Furthermore, the need for a sea base is based on a strategic concern of lack of a land base to operate from.

Core competencies also can have an effect on the architecture. The U.S. Marine Corps has a well established method of amphibious warfare which suits their particular force makeup and training. As a result, many proposed sea base architectures are based on their current operational doctrine and force composition. Additionally, there are several legacy elements in the military which impose compatibility constraints. Several pieces of equipment have significant impact due to their weight, size, or magnitude. For instance, the main battle tank used by the U.S. Army and U.S. Marine Corps are over twice as heavy as any other piece of equipment. Since the difficulty in transferring equipment at sea grows exponentially with weight, this imposes a significant design and architecture constraint.

The environment, both physical and competitive, also impacts the architectural design. The sea is generally an unforgiving place to operate and thus drives certain interfaces and eliminates some designs as unsuitable in rough conditions. For instance, using an air cushioned vehicle in the open ocean during rough seas would be extremely dangerous due to limited stability. The competitive environment also requires that the various platforms incorporate some form of protection from enemy attack or operate with other forces within the protection of Sea Shield

The final upstream impact, technology, has the greatest impact on architecture. The desirability of having a 100 knot ship capable of carrying 10,000 tons of equipment is certainly a laudable goal, but technically infeasible. Changing technology in terms of better materials and control systems, however, are what enable certain capabilities such as transferring a main battle tank between ships in heavy seas to be a possibility that would have been considered infeasible just a few years ago.

The downstream side also influences the architecture. Manufacturability is a major concern for the design and imposes concerns. For example, the desire to build in the U.S. puts additional complexity into the design such as whether a new shipyard needs to be built or to constrain the design to fit into existing shipyard facilities. Locations where the sea base will operate and be maintained also impose downstream architecture constraints such as draft limits.

The expected operations also should impact the architecture. The sequence of operations could be considered established by the upstream influences such as current doctrine and thus forces the architecture into a certain set of possibilities. However, doctrine is constantly changing as new capabilities are recognized thus the downstream operations may alter or make moot some of the upstream influences.

The dynamic behavior is also a very important factor. Ranges between the various nodes (U.S., sea base, objective), battle attrition, sea conditions, and other uncertainties can have a major impact on the effectiveness of different architectures. Thus these uncertainties must be considered to ensure a robust design is selected.

Finally, as with most systems, cost must be considered as a critical factor. Not only must the architecture be designed to minimize cost, but cost may also impact the problem statement as the design progresses and the achievement of certain goals might exceed the allocated budget. Thus, both budgets and designs must be flexible to reallocate resources.

Functional Description

With a well defined problem statement and concept, the architecture can examine the functionality in more detail. From a functional standpoint, the sea base needs to perform many tasks in order to properly deploy the combat force. The first level processes were obtained from the Department of Defense Joint Integration Concept for Sea Basing and are summarized below:

Deliver forces that will reside on or be supported by the sea base
Assemble integrated joint forces and equipment
Beach combat ready integrated joint forces
Sustain selected forces from the sea base
Reconstitute joint force capabilities for rapid reemployment

Zooming into the process of deploying reveals the first level processes. A further zoom shows the subprocesses which represent additional functions. This process zoom is shown in Figure D4. Most of the processes are linear in time, thus the arrows show the order the processes are normally taken. For instance, the joint force is delivered before it is assembled.

Most of the processes, and their subprocess, act on the operand of deploying, namely the joint combat force. The sustaining process, however, acts on the joint combat force, yet its subprocesses actually act on the consumable supplies needed by the joint combat force. Thus each process represents part of a function. Each of these functions is described in more detail below.

The first function involves taking the joint force from the U.S. to the sea base. This includes the troops, their equipment, and immediate supplies. These forces can then either reside on the sea base relatively permanently, such as logistics or command personnel or can be located on the sea base temporarily, like most of the fighting force, but still supported from the sea base. Within this delivering process, there are the 2nd level processes. The troops and equipment must be gathered together, stored for the journey, moved to the sea base, and then transferred to the

next platform. Concurrent with these processes, the joint combat force must be protected not only from enemy forces, but also from environmental conditions such as weather and seas.

The second function involves assembling or connecting the forces, equipment, and supplies together. This requires transporting or gathering elements of the force within the assembling warehouse. Some element of the force will be prepared which may involve training, calibration, fueling, etc. depending on the force. From there the force must be stored before being transferred to the next platform.

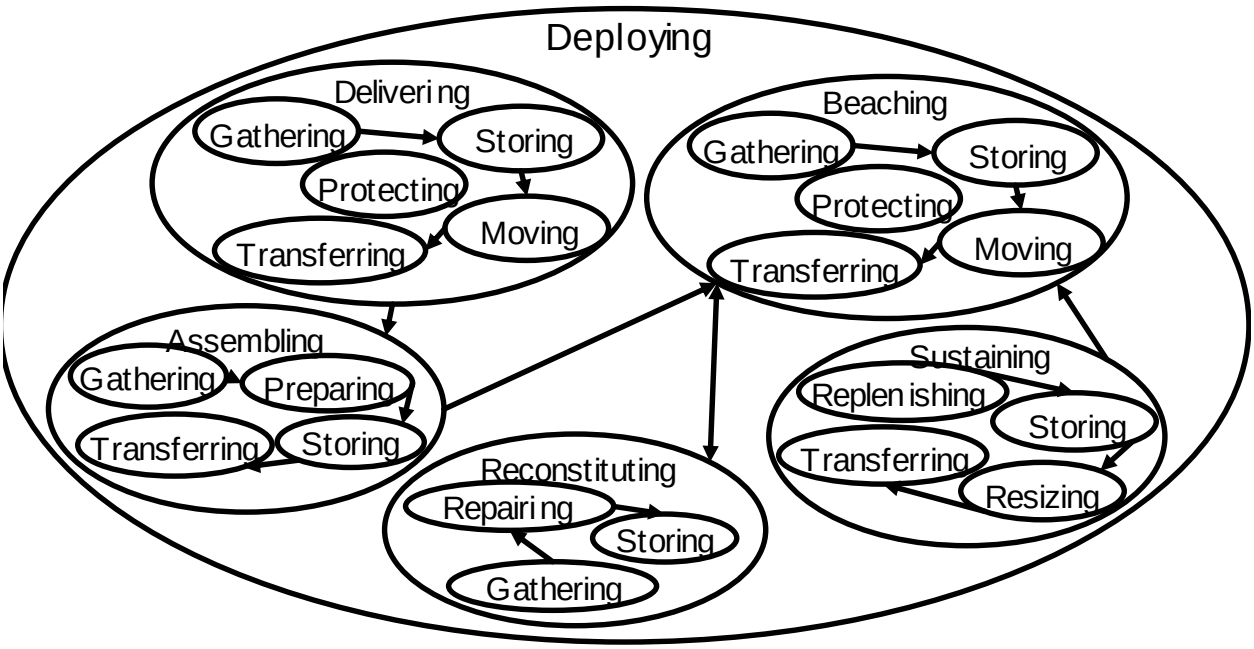


Figure D4. Sea Base Process Zoom

The third function transports the assembled force to the objective. So as not to confuse this function with the first function, it will be referred to as beaching. This does not necessarily mean that the force much be placed on a beach. The force could also be sent to an objective further inland using a helicopter for instance. The subprocesses for beaching are exactly identical to those of delivering since they perform the same basic function. They are considered separate because each is influenced by different interfaces and upstream influences, such as environment, which significantly impacts the ultimate form.

The fourth function is sustaining the force indefinitely. Sustainment in this context implies providing consumable supplies such as water, food, ammunition, and fuel to the joint force. These supplies are typically stored in bulk and periodically resupplied from replenishment ships. The bulk supplies must be resized so they can be carried by individual force members. Finally, the supplies must be transferred to a platform capable of delivering them to the force.

The final level 1 function is reconstitution. This involves taking troops back from the beach. The force is repaired and reorganized (or gathered) to prepare for a different objective. The force is then stored until it is transferred to a platform to take it once again to the beach.

Form Decomposition

The sea basing concept described above involved three different platforms, a transporter, a warehouse, and a lighter. Between each of these platforms has an interface. Since these interfaces are critical to the form design of each platform they are treated separately. Finally, there is also a replenishment interface to satisfy the resupply function of indefinite sustainment. Each of these pieces can be further broken down into subelements as shown in Figure D5. Two additional elements, the ESG and replenishment ships, are shown to indicate their auxiliary role in sea basing in providing some of the same functions to different joint combat forces.

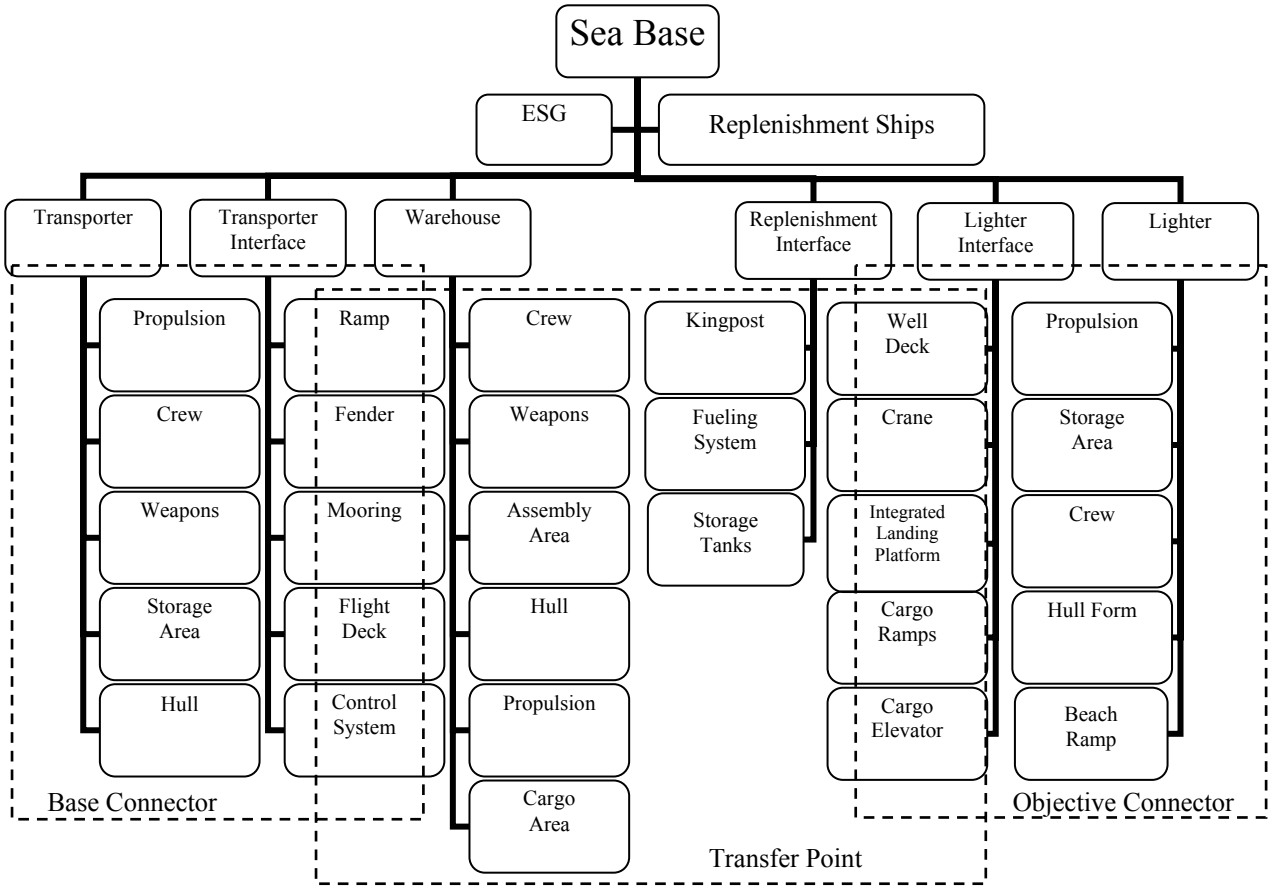


Figure D5. Two Level Form Decomposition of Sea Basing

Figure D5 is drawn as a structural/topological decomposition where each of the three platforms is identified by a dashed line. One characteristic is the overlap at the interface forms. Some of the parts for each of these forms will be on each platform, but at this level of decomposition, the form elements are in general shared by the two platforms. This form decomposition also has elements of operational decomposition indicating how the force moves over time from the base connector through the transporter interface to the transfer point through the lighter connector to the objective connector. However, each of the platforms perform many of the same functions, thus the form decomposition could be reorganized to relate more directly to function. In this case, the transfer point platform may have transporter elements, as the platform must move into the sea base area and warehouse elements where it stores and assembles the force.

Figure D6 shows how the forms and functions could be combined. In this diagram it is less clear where one platform starts and another ends since many of the form elements actually exist on several of the platforms.

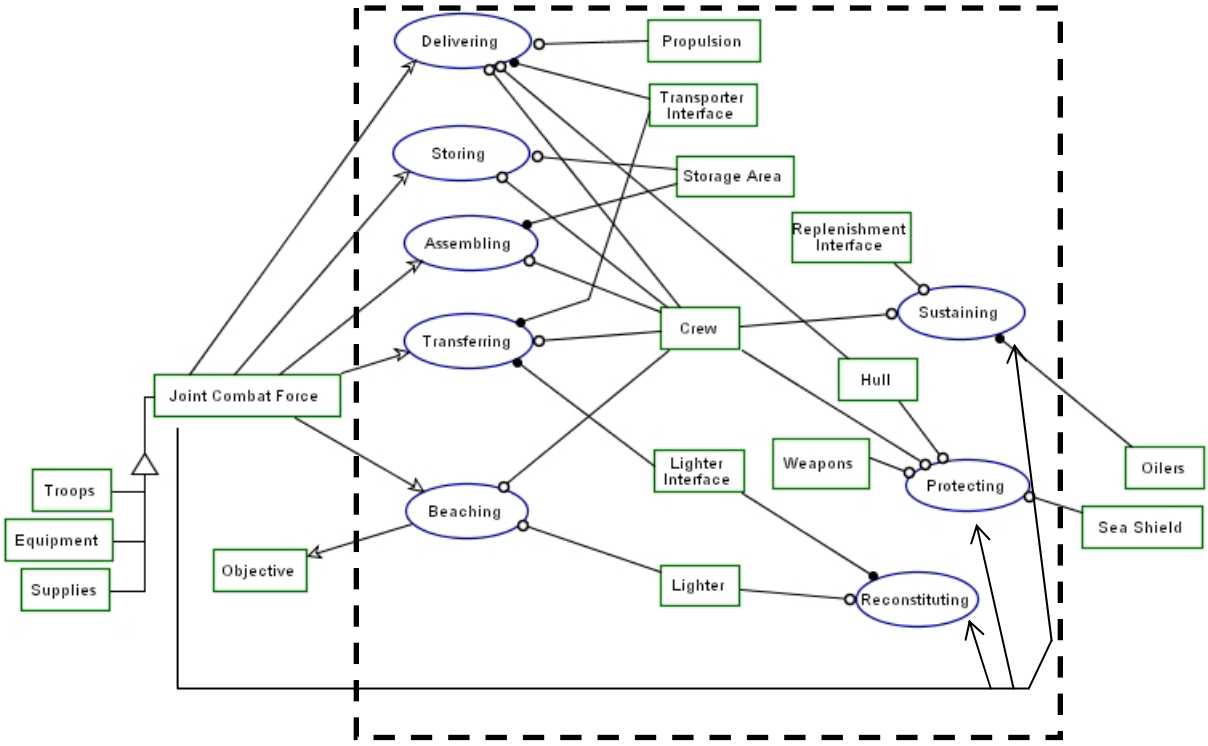


Figure D6. Sea Basing Form Coupled with Function

Operational Description

Based on the three platform sea base concept and the form given in the previous section, it is important to understand how the subsystems operate together. The best way to describe the operations of the sea base is to describe a scenario and demonstrate how the sea base would be utilized.

Imagine "Country Orange", henceforth referred to as Orange, has declared war on the U.S. and is supporting terrorist attacks on U.S. property. The countries bordering Orange have leaning toward Orange, but remain neutral. Therefore, basing rights are unobtainable and escalating the conflict by invading neighboring countries is unacceptable. The objectives are strategic "centers of gravity" such as terrorist training camps and military bases directly. Unfortunately, these "centers of gravity" are located within civilian areas and change locations rapidly. Thus, a ground force is needed to ensure the objectives are destroyed while minimizing collateral damage. Since basing rights are not obtainable from neighboring countries, the decision is made to employ the sea base. Equipment and supplies are loaded onto the transfer ship(s) based on the expected needs. Additional equipment and supplies are gathered and placed on sealift base connectors. These ships begin their movement toward the Orange Sea. In the U.S., troops prepare themselves and their families for the upcoming battle before getting onto airlifting base connectors. The transfer ship arrives in the Orange Sea, where other warships have established sea control. The troops aboard the airlift connectors and the equipment aboard

the sealift connectors are transferred to the transfer ship via transporter interfaces such as the flight deck and ramps in conjunction with the other elements.

Aboard the transfer ship, combat support personnel are preparing the equipment for battle and the troops join their equipment. The troops and equipment remain on the transfer ship until the time to attack the objective comes. At this point, the joint force is moved from the assembly area through the lighter interface to board objective connectors. Once to connector is fully loaded and secured, it proceeds to the objective where it delivers the joint force.

As the battles progress, the joint force uses up its supplies. Some of the objective connectors not being used to transport equipment and troops are used to resupply the joint force wherever they happen to be located. The transfer ship, also periodically receives new supplies, such as oil, from replenishment ships.

After an objective is destroyed, the joint force is gathered by the objective connectors and returned to the transfer ship. The equipment is repaired, troops rested, and wounded troops receive medical attention. Replacement equipment and troops can also be augmented and gathered with the returning force. Once the force is reconstituted and battle ready, they are delivered to another objective in a similar method as described above.

Figure D7 shows a simplified topological diagram of the sea base. It shows the activities that go into and out of the sea base and what each of the three platforms of the sea base (base connectors, a transfer ships, and objective connectors) operate on. There are significant things that this diagram does not show. The cargo and supplies have many legacy elements to them in their size, weight, and/or amount. The troops, equipment, and supplies must come from somewhere, either the U.S. or a land base. The distance and travel time also impact the form of the base connector in order to meet the system goals.

Critique of Architecture

It is important to note that the sea base as described in this paper does not exist. As discussed in the conceptual development section, there are numerous architecture options which may achieve the same objective as sea basing. Most of the other architectures either already exist in some form or are proposed as alternatives to the architecture described in more detail The argument that led to sea basing is based on several assumptions such as complete lack of allies, which many military strategists find unlikely while the U.S. remains capable of force projection. Even if there is a need for sea based forces, there are several architectures that could be used.

The U.S. Army and U.S. Air Force are exploring using a single platform to take a force directly to the objective using high speed catamarans or aircraft as well as existing landing craft. The assembly would be done before or during the transportation. This architecture is generally considered inefficient because of large transportation fuel costs and the number of trips required to transport a sufficiently sized force.

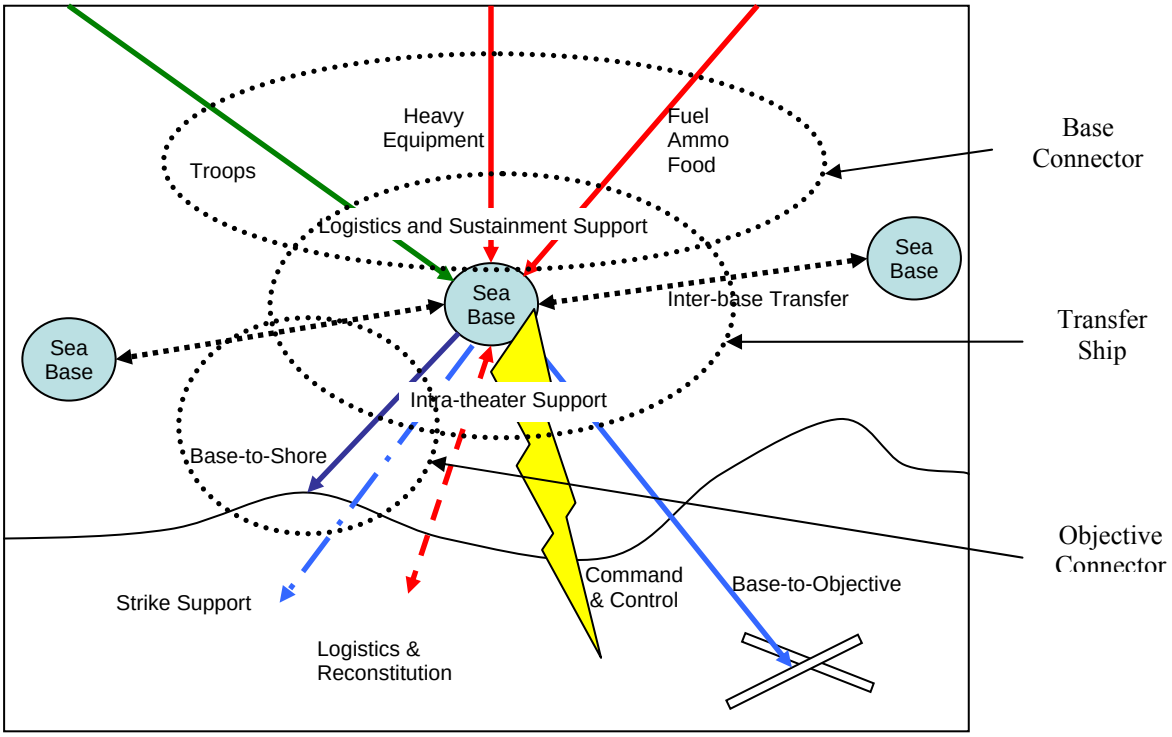


Figure D7. Sea Base Operations

The U.S. Navy and U.S. Marine Corps are currently exploring a two platform option called MPF(F) which essentially eliminates the need for the base connector and transporter interface by carrying the entire force to the theater on a squadron of ships. The problem with this method is robustness. The MPF(F) is designed with one size force in mind and thus would be less suitable if a different force is needed.

Another two platform solution uses a virtual transfer ship by combining the transporter interface and lighter interface. In this construct, the base and objective connectors would interface with each other and be continuously cycle between the three locations (U.S., sea base, and objective). This requires having objective connectors which can operate independently since there is no transfer ship that can carry them to the sea base. This causes additional design constraints on the objective connector making it less efficient.

The three platform option presented above provides additional flexibility over the MPF(F) since different forces can be transported to the sea base by the base connectors both before and during the conflict. It is also a more feasible and efficient alternative to the virtual transfer ship or the single platform options. This architecture is most similar to Just-In-Time logistical processes currently in business vogue. The transfer ship acts as like a Wal-Mart store. The objective connectors are shoppers who take the product off the shelf. The base connectors are similar to the periodic shipments received from suppliers which bring only the product desired by the customers. Extra product is kept to a minimum and is flexible to the needs of the customer.

The negative to the three platform architecture is the potentially greater acquisition cost. A minimum of three different platforms implies three different designs and sets of development costs. Economies of scale might be achieved from building more of fewer sets of platforms. These economies of scale are less in the three platform architecture. Operational costs may also be greater for the three platform architecture over other architectures since there could be more

transits between the U.S. and the sea base. A final negative consequence of this architecture may be in delivery time. Larger travel distances between the U.S. and the sea base increase the time required to deliver the complete force to the sea base which could impact mission accomplishment.

Conclusion

An issue with every architecture option is bias. The proponents of the various architectures may have certain biases which tend to accentuate the advantages while avoiding the difficulties and problems. Since no architecture currently exists, the desired tradeoff between flexibility, cost, efficiency, and delivery time has not been properly assessed which leaves a great deal of room for argument. Changing technologies also add another dynamic effect. The feasibility of some of the enabling technologies also creates an element of risk but also present opportunities. The major concern is that many of the high level architectural decisions lock in design choices which will impact a system which will likely be around for the next 30 to 40 years. Ultimately, the best choice is to continue exploration of the conceptual options to determine the necessary technology investment and better define the potential opportunities of sea basing while continuing collaboration to reduce any architectural bias among the system architects.

Page Intentionally Left Blank

Appendix E: Genetic Algorithm System Level Optimizer

Unlike traditional gradient based numerical optimization methods, GAs are well suited for discrete variables that predominate when dealing with SoS (e.g. number of each system involved in SoS, type of interface, etc.) GAs also perform well in nonlinear design spaces where other algorithms may have problems. Furthermore, GAs are able to work in large infeasible design spaces which hinder many gradient based optimization techniques. Finally, because of the randomness inherent in GAs, they perform better at locating global optimum points than gradient based techniques which require multiple optimizations from different starting points to improve the chance the global optimum is captured.

The major drawback with GAs is that they are population based, which means they are typically more computationally intensive than other optimization algorithms. The need to perform gradient based analysis from multiple starting points reduces the effect of this disadvantage. Furthermore, as a population based method, the GA is able to present a range of possible solutions which is especially useful in multiobjective design.

A GA was chosen as the system level optimizer for several reasons. First, the problem is nonlinear in both the performance model and the subsystem model constraints. The performance model is a discrete time based simulation which provides a significant nonlinearity. Second, the problem formulation includes a large number of integer variables making this a mixed-integer problem difficult to be solved using traditional numerical optimization methods. Finally, for a multiobjective problem, a population based approach can identify a POF in one optimization run. Since the shape of the POF is not known, the GA provides a robust technique which avoids some of the difficulties with typical weighed sum approaches to multiobjective problems.

The GA used in this research was GOSET Version 1.03 developed at Purdue University by Professor Scott Sudhoff. This algorithm was written in MATLAB, provides mixed-integer and multiobjective capabilities, and can be tailored to the specific problem. It uses many standard GA techniques to improve the population and follows a program flow described in Figure E1. The following discussion will describe how the GA was implemented for this problem. Goldberg (1989) and Deb (2001) are excellent sources for further information and greater details. Additionally, GOSET provides additional functionality not utilized in the research, described in the GOSET Manual (Sudhoff & Lee, 2004).

The GA begins by creating an initial population of individuals with randomly generated genes between specified boundaries. For this problem, the genes were the system level variables with bounds described in Table E1. GOSET is able to accommodate both continuous and integer variables, however in order to speed the fitness evaluation process, all variables were scaled and held to integer values. This constraint limits the ability to accurately represent the actual POF, but the design space is sufficiently large and the design tolerances wide enough at the preliminary stage to make this a small sacrifice.

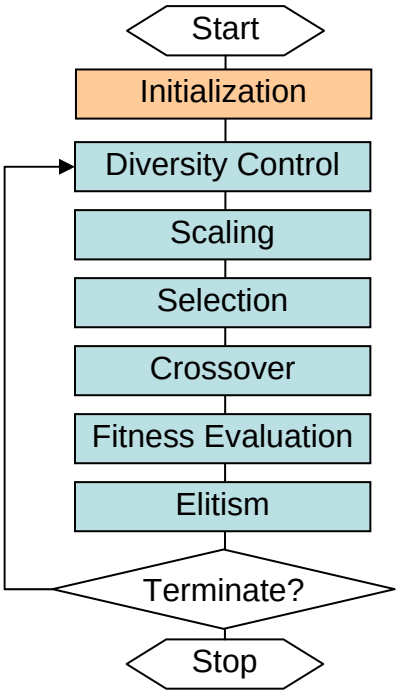


Figure E1. Flowchart of GOSET 1.03 Genetic Algorithm

Table E1. Genetic Algorithm Variables

Gene	Min Value	Max Value	Type
Cargo Speed	20	50	Integer
Cargo Number	1	15	Integer
Cargo Capacity (x125LT)	2	11	Integer
Cargo Capacity (x1000sqft)	1	25	Integer
JOAShip Number	1	8	Integer
JOAShip Capacity sqft (x5000)	10	80	Integer
JOAShip Capacity Personnel (x100)	10	50	Integer
JOAShip Sites air	1	15	Integer
JOAShip Sites well	1	3	Integer
JOAShip Sites side	1	2	Integer
JOAShip Lighter LCAC	0	8	Integer
JOAShip Lighter MV-22	0	20	Integer
JOAShip Lighter CH-53	0	20	Integer
JOAShip Lighter SeaBAC	0	8	Integer
Lighter Speed	20	50	Integer
Lighter Capacity (x25ton)	3	12	Integer
Lighter Length (x10)	2	18	Integer
Lighter L/B Ratio (/10)	18	22	Integer

Each individual of the initial population is evaluated for its fitness. The fitness function includes running each of the subsystem optimizers to evaluate the platform feasibilities and displacements. The function then determines the objective values by running the cost and performance models. Constraints from the subsystem optimizers are applied to each objective value as a penalty function to develop a fitness value described in Equation (E1).

$$J_f = J_o * \sum .0001 * \max(g^i(x),0) / w^i \qquad \text{(E1)}$$

where J_f is the fitness value
J_o is the objective value
$g^i(x)$ is the constraint value returned by subsystem optimizer i
w^i is a scaling factor for subsystem i to bring the constraint to O(1)

Table E2. Genetic Algorithm Constraints

Output Variable	Normalization Value	Reason
HSC Feasibility Error	None	Already normalized internally
HSLS Feasibility Error	225	Max error in previous trials
ITS Feasibility Error	235	Max error in previous trials

After evaluation, the fitness of the population undergoes diversity control and scaling. Diversity control spreads solutions across the design space to minimize speciation, i.e. where a single or small subset of fit individuals dominates the population. Scaling, on the other hand, helps prevent the early domination of extraordinary individuals, while encouraging healthy competition between individuals during later generations.

The ultimate goal is to obtain a single fitness value for each individual to be used for selection and crossover. First, the various objectives are summed together using a randomly determined weighting to determine a fitness value. Alternatively, a specific weighting could be assigned in order to preferentially select for a certain objective; however, this was not done for this research.

With a single fitness value, diversity control is applied. The diversity control used in this research assumes that individuals with similar gene values have similar weighted sums of their gene values. Using a random weighting, the genes are combined and then divided into a random number of bins between 50% and 200% of the population size. The fitness of each individual is divided by the number of individuals in that individual's bin. Because of the initial assumption on weighted sums of the genes, two very different genes could have similar weighted sums. Therefore, the algorithm is performed several times with different weighting schemes and the highest penalty is ultimately applied. The weighted sum diversity control was used due to the shorter computational time versus a more systematic approach such as normalized distance between individuals.

Before applying the diversity penalty, the fitness values are scaled using linear mapping to assign values between 1 and 101 based on the initial fitness value. Linear mapping was used because of problems resulting from standard and median scaling. These methods maintain the mean and median fitness respectively and truncate fitness values at zero. However, when dealing with highly infeasible regions with only a few feasible solutions, many of the individuals can end up with a fitness of zero which can cause problems later with selection and mutation.

After applying the scaling and diversity control penalty to the fitness value, a roulette wheel selection technique is used to select the population for crossover. This technique calculates a ratio of each individual's fitness to the total population fitness (the sum of each individual's fitness). The ratio represents the percentage chance of selection and ensures those individuals with the highest fitness are preferentially selected for crossover. During crossover, groups of two individuals, referred to as parents, exchanges genes about a random crossover point to create two new "children". In total, 60% of the population is replaced by children.

The mutation operations also can produce new individuals. Selection and crossover could cause some possible genes to be lost within the population, so mutation is included to help avoid this problem. Because all the variables in this problem are integers, only integer mutation is used where each gene has a .8% probability of being assigned a new value within its defined limits.

Crossover and mutation were used to create new individuals. Elitism on the other hand is used to retain individuals. This is especially useful in multiobjective optimization since elitism allows nondominated solutions to be retained. After evaluating the new individuals, the nondominated individuals from both the old and new population are kept. They randomly replace up to 50% of individuals in the new population. If the number of nondominated individuals exceeds 50% of the population, the nondominated individuals are selected based on diversity controls.

Once a new population is completed, if the desired number of generations has been evaluated, the algorithm terminates, otherwise, the process begins again.

Page Intentionally Left Blank

Appendix F: Genetic Algorithm Setup Code

```
function [fP,GAS] = ship_optimizing()
%
% This routine is the system level setup that calls GOSET
%
% Created by Robert Wolf – Jan 2005
%
% [fP, GAS] = ship_optimizing
%  fP       = Final population
%  GAS      = Genetic Algorithm Statistics
%
close all
clear all
format compact
%*********************************************
% User Defined Nonoptimized Constant Parameters (Assumptions)
%*********************************************
% Data setup for Fitness function
Param.NumESG=2;
Param.MV22=0;
Param.CH53=0;
% Mission Performance Model Parameters
Param.NumITS=1;         % Number of ITS Variants (Listed first in JOAShips)
Param.CargoSize=3;      % Size of Cargo Variables (Tons, SqFt, People)
Param.SiteSize=3;       % Sites (Air, Well, Side)
Param.PlotFreq=5;       % Plotting Frequency (in Time Steps)
Param.PlotFlag=0;       % 0 - No plots
Param.TimeSteps=30;     % # time steps per hour (12 = 5 min per step)
Param.Distance=[25,110,500,2000];% Surf Obj Dist, Air Obj Dist, AB Dist
%   From MPF(F) AoA, The daily sustainment requirements for the MEB are:
%   Food; Water; Fuel; Ammo (tons, sq.ft., people)
Param.DailySustain=[15, 29, 0; 169, 337, 0; 254, 141, 0; 250, 250, 0];
Param.DailySustain=[12.4, 39, 0; 60.5, 150, 0;128.4,70, 0; 96.7, 100, 0];
Param.DaysSustain=5;       % Number of Days Sustainment on ITS
Param.AAAVSize=[37.25 358.2 20];  % Size of AAAV
Param.AAAVSpeed=20;               % Speed of AAAV
Param.SupportElement=[0 0 760];% From MPF(F) AoA p.10 w/o air wing
Param.FlightHours=24;        % Hours per day air lighterage allowed to fly
Param.PrepTime=3;            % In hours
Param.DayMax=3;           % Max number of days until simulation termination
%
% The following code sets up variables for GOSET
numobj=2;    % Number of Objectives (Cost, DeliveryTime
GAP=gapdefault(numobj);  % Get default values of GOSET parameters
GAP.fp_ipop=5;           % Size of initial population
GAP.fp_npop=5;           % Size of subsequent populations
GAP.fp_ngen=2;           % Number of generations
GAP.fp_obj=0;            % objective to optimize (0 for multiobjective)
GAP.op_list=[];          % list of objectives to make objective plots for
GAP.pp_list=[2 1];       % list of 2 or 3 parameters to be used in Pareto
plot
GAP.pp_xl='Cost';        % x-axis label
GAP.pp_yl='DeliveryTime'; % y-axis label
GAP.pp_axis=[-80 0 -48 0]; % axis limits for Pareto plot
GAP.rp_itd=0;            % No time report
```

```
GAP.sc_alg=4;            % Scaling algorithm (linear mapping)
GAP.sc_kln=101;          % Scaling parameter (size of linear map)
GAP.mc_alg=1;            % Crossover algorithm (single point)
% Zero the non-integer mutation rates
GAP.mt_ptgm=0;
GAP.mt_prgm=0;
GAP.mt_pagm=0;
GAP.mt_prvm=0;
GAP.mt_pavm=0;
GAP.mt_savm=0;
%
% Setup Gene bounds and type(i.e. integer)
%      minval  maxval  type
Genes=[ 20      50      1;...   % Cargo Speed
        1       15      1;...   % Cargo Number
        2       11      1;...   % Cargo Capacity (x125LT)
        1       25      1;...   % Cargo Capacity (x1000sqft)
        1       8       1;...   % JOAShip Number
        10      80      1;...   % JOAShip Capacity sqft (x5000)
        10      50      1;...   % JOAShip Capacity Personnel (x100)
        1       15      1;...   % JOAShip Sites air
        1       3       1;...   % JOAShip Sites well
        1       2       1;...   % JOAShip Sites side
        0       8       1;...   % JOAShip Lighter LCAC
        0       20      1;...   % JOAShip Lighter MV-22
        0       20      1;...   % JOAShip Lighter CH-53
        0       8       1;...   % JOAShip Lighter SeaBAC
        40      70      1;...   % Lighter Speed
        3       8       1;...   % Lighter Capacity (x25ton)
        2       18      1;...   % Lighter Length (x10)
        18      22      1];     % Lighter L/B Ratio (/10)
minvals = Genes(:,1)';
maxvals = Genes(:,2)';
type = Genes(:,3)';
chrom_id = ones(size(type)); % All on the same chromosome

[fP,GAS]= gaoptimize(@Seabase_FitnessV2,numobj,0,Param,minvals,maxvals,...
  type,chrom_id,GAP,[],[],[]);
```

```
function f = SeaBase_fitness(GAInput,Param)
%
% This function is called by the GOSET GA algorithm (gaoptimize)
% The purpose of this function is to evaluate a set of genes and determine
%   their fitness
% This function also calls the 2nd level routines
%
% Created by Robert Wolf – Jan 2005
%
% Output: f = fitness value
% Input : Input vector that consists scaled values of
%    GAInput(1-4)   = CargoShip (Speed, #, Cap(LT, sqft))
%    GAInput(5-14)  = JOAShip (#,Cap(sqft, Pers),Sites(x3),Lighter(x4))
%    GAInput(15-18) = Lighter (Speed, Cap(LT), Length, Beam)
% Param : Non-optimizing parameters structure variable
%    .NumESG      = Number of ESG (consists of LPD, LHA, LSD)
%    .MV22        = Additional MV22 in scene (perhaps from carrier)
%    .CH53        = Additional CH53 in scene (perhaps from carrier)
%    Other variables for Mission Simulation
%
% Make sure the input variables are real numbers
if any(~isfinite(GAInput))
    load SimPerfData2
    Cost=999
    output.DeliveryTime=999
    feasible=1e-12
else
%
% Setup the Variable structures
[CargoShips,JOAShips,Lighter]=ThesisShipData(Param); % Get default values
[ESGCargo]=CargoListESGMEB; % Cargo aboard ESG
[SurfCargo]=CargoListSurfMEB; % Non-ESG Surface Objective Landing Team
[AirCargo]=CargoListVertMEB; % Non-ESG Air Objective Landing Team
AirBLT=1;SurfBLT=1; % Number of Landing Teams
CargoList=[ESGCargo;repmat(AirCargo,AirBLT,1);repmat(SurfCargo,SurfBLT,1)];
%
CargoShips(1).Speed     = GAInput(1);
CargoShips(1).Number    = GAInput(2);
CargoShips(1).Capacity  = [GAInput(3)*125, GAInput(4)*1000, 200];
CargoShips(1).LoadTime  = ceil(4*GAInput(3))/4; % ~125LT per hour
CargoShips(1).UnloadTime= CargoShips(1).LoadTime;
%
JOAShips(1).Number      = GAInput(5);
JOAShips(1).Capacity    = [30000, GAInput(6)*5000, GAInput(7)*100];
JOAShips(1).Sites       = [GAInput(8:10)];
JOAShips(1).Lighter     = [GAInput(11),0,GAInput(12:14)];
JOAShips(1).LighterB    = GAInput(17)*10/(GAInput(18)/10);
JOAShips(1).LighterL    = GAInput(17)*10;
%
Lighter(5).LSpeed       = GAInput(15)*.8;
Lighter(5).USpeed       = GAInput(15);
Lighter(5).Capacity     = [GAInput(16)*25,.45*GAInput(17)*GAInput(18), 24];
Lighter(5).LoadTime     = .5*GAInput(16)*25/75/((GAInput(18)/10)/50);
Lighter(5).UnloadTime   = Lighter(5).LoadTime;
Lighter(5).Length       = GAInput(17)*10;
Lighter(5).Beam         = GAInput(17)*10/(GAInput(18)/10);
Lighter(5).Sites        = [0,ceil(Lighter(5).Beam/50),0];
```

```
%
% Check previous iterations to see if this is a duplicate
%
load SimPerfData2; % Location of previous optimization results
SimResults=[];
for i=1:1:size(SimPerfData2,1)
    if all(SimPerfData2(i,1:18)==GAInput(1:18))
        SimResults=SimPerfData2(i,19:20);
        Output.DeliveryTime=SimPerfData2(i,20);
        Cost=SimPerfData2(i,21);
        C1=SimPerfData2(i,22);
        C2=SimPerfData2(i,23);
        C3=SimPerfData2(i,24);
        break
    end
end

if isempty(SimResults) % No identical previous iteration?
% Evaluate the 2nd level optimizers
% Optimize the CargoShip
%   Input:  CargoShip variables (Speed, Num, Cap(ton), Cap(sqft))
%   Output: C1     = Subsystem error (0 if feasible)
%           Output1= Displacement
try % If an error occurs, keep from terminating the optimization
[C1,Ceq1,Output1]=HSC_CO_Optimizer(CargoShips(1),Param);

if JOAShips(1).Lighter(5)~=0 % HLSL present?
% Optimize the Lighter
%   Input:  Lighter variables (Speed, Cap(ton), Length, Beam)
%   Output: C2     = Subsystem error (0 if feasible)
%           Output2= Displacement
[C2,Ceq2,Output2]=SeaBAC_CO_Optimizer(Lighter(5),Param);
else % No HLSL, then skip subsystem and remove coupling constraint
    C2=0;Ceq=0;Output2=0;JOAShips(1).LighterB=0;JOAShips(1).LighterL=0;
    GAInput(15:18)=0;
end

% Optimize the JOA Ship
%   Input: JOAShip variables and Lighter Size
%   Output: C3     = Subsystem error (0 if feasible)
%           Output3= Displacement
[C3, Ceq3, Output3]=ITS_CO_Optimizer(JOAShips(1),Param);
Output3=Output3*10000; % Unscale the displacement

% Check performance if all subsystems feasible
%   Input (JOAShip, CargoShip, Lighter)
%   Output (DeliveryTime)
if and(and(C1==0,C2==0),C3==0)
    Output=MissionSim5(CargoShips,JOAShips,Lighter,CargoList,Param);
else
    Output.DeliveryTime=24*Param.DayMax;
end

% Check cost
%   Input (Number of ships, and costs)
%   Output (system cost)
```

```
k1=4;k2=8;k3=1;
Cost=k1*Output1*CargoShips(1).Number+...
     k2*Output2*JOAShips(1).Number*JOAShips(1).Lighter(5)+...
     k2*153.3*JOAShips(1).Number*JOAShips(1).Lighter(1)+... % for LCAC
     k3*Output3*JOAShips(1).Number;
Cost=Cost/1e4;

catch % if there was an error, then set special errors and outputs
    Output.DeliveryTime=24*Param.DayMax
    Cost=max(SimPerfData2(:,21));
    C1=999;C2=999;C3=999;Output1=999;Output2=999;Output3=999;
    disp('error')
end
% Record new det of iteration data for file
SimPerfData2(size(SimPerfData2,1)+1,:)=[GAInput, [Output.DeliveryTime],...
  Cost, C1, C2, C3, Output1/1000, Output2/100, Output3/10000];
end
%
% Apply penalty constraints
feasible=1;
Constraint=[C1/1,C2/100,C3/100];
for i=1:1:size(Constraint,2)
    if Constraint(i)>0
    feasible=feasible*.0001/max(Constraint(i),.001);
    end
end

end

% Define Objectives
f=[-max([Output.DeliveryTime])/feasible;...
   -Cost/feasible]; % Multiobjective output must be a column vector

% Save iteration data file
save SimPerfData2 SimPerfData2;
```

```
function [CargoShips,JOAShips,Lighter]=ThesisShipData(Param)
%
JOAShips(2).Name='LPD-17';
JOAShips(2).Number=Param.NumESG;
JOAShips(2).Lighter=[2,0,0,0,0];
JOAShips(2).AAAV=12;
JOAShips(3).Name='LSD-41';
JOAShips(3).Number=Param.NumESG;
JOAShips(3).Lighter=[0,2,0,0,0];
JOAShips(3).AAAV=12;
JOAShips(4).Name='LHD';
JOAShips(4).Number=Param.NumESG;
JOAShips(4).Lighter=[3,0,10,3,0];
JOAShips(4).AAAV=12;
JOAShips(5).Name='Carrier';
JOAShips(5).Number=1;
JOAShips(5).Lighter=[0,0,Param.MV22,Param.CH53,0];
JOAShips(5).AAAV=0;
%
Lighter(1)=struct('Name','LCAC',...
    'LSpeed',   35,...
    'USpeed',   40,...
    'Capacity', [75,1800,24],...
    'LoadTime', .6,...
    'UnloadTime',.5,...
    'Air'       ,0,...
    'Sites',    [0,1,0],...
    'Attrition' ,0);
Lighter(2)=struct('Name','LCU',...
    'LSpeed',   10,...
    'USpeed',   15,...
    'Capacity', [200,3000,100],...
    'LoadTime', .85,...
    'UnloadTime',.75,...
    'Air'       ,0,...
    'Sites',    [0,0,0],...
    'Attrition' ,0);
Lighter(3)=struct('Name','MV-22',... % STOM CONOPS p.5-3
    'LSpeed',   135,...
    'USpeed',   240,...
    'Capacity', [5,410,24],...
    'LoadTime', .25,...
    'UnloadTime',.25,...
    'Air',      1,...
    'Sites',    [1,0,0],...
    'Attrition' ,0);
Lighter(4)=struct('Name','CH-53',... % STOM CONOPS p.5-3
    'LSpeed',   125,...
    'USpeed',   135,...
    'Capacity', [13.4,600,36],...
    'LoadTime', .25,...
    'UnloadTime',.25,...
    'Air',      1,...
    'Sites',    [1,0,0],...
    'Attrition' ,0);
```

Appendix G: Performance Simulation Model

When analyzing how the sea base performs, it is important to understand the greater system dynamics of the system of ship systems. The ultimate goal of the sea base is to provide adequate logistical throughput to deploy and sustain a U.S. Marine Corps 2015 MEB sized unit. This particular unit size is not special in that a smaller or larger unit could also be used, but was chosen as the baseline due to the availability of equipment information and sustainment requirements and because the primary sea base customer is the U.S. Marine Corps.

Simulation Concept

The sea base model is set up with three sets of nodes with lines indicating the flow path. Figure G1 represents a nodal diagram of the setup.

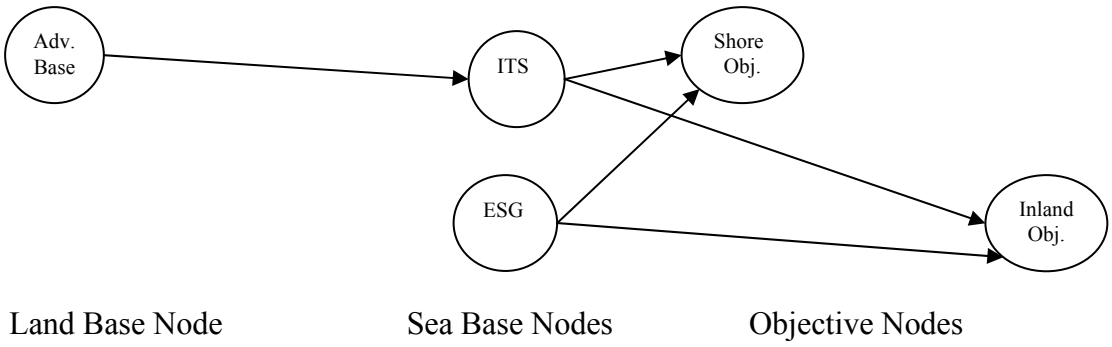


Figure G1. Nodal Diagram of Sea Base

The first node is the Advance Base (AB) which is located some distance from the operating area. While referred to as an advance base, it could also be located in the U.S. It is considered a secure place to store and maintain equipment and personnel. The AB could be multiple locations (i.e. equipment at one AB and personnel at another) and is assumed to be an infinite source of logistics. In other words, no attempt is made to plan the logistics from the point of departure from the U.S. to the AB. This means the equipment is prepositioned, airlifted, or sealifted to the AB prior to being required by the sea base. From the AB the joint force must be transported to the sea base.

The second node is the sea base itself. This consists of some number of ESGs made up of an amphibious ship from each of three ship classes: LSD-41, LPD-17, and LHD-1. Specifying these ships is arbitrary as other ships including new designs such as the LHA(R) could also be used. The sea base also contains some number of ITS which act as the conduit for the personnel, equipment, and supplies for the joint force. The ITS may or may not carry cargo to the sea base based on its warehousing capacity. However, the ITS must have interface positions, both surface and air, to allow the transfer of the joint force. Each ship at the sea base, ESG and ITS, is assumed to start with a full cargo load to support the operation. This cargo load may not contain the full MEB and additional deliveries from the AB may be required.

The third node is the objective. This is the final destination for the joint forces and could be located on the shore or some distance inland. Forces must be transported from the sea base to the objective via air or sealift from either the ships of the ESG or ITS. The distances between the

sea base and the objectives can be different and could vary during the simulation as both the sea base and the objective change position. Each objective is assumed to be an infinite sink in that cargo can be delivered in any quantity without problems. The infinite sink assumption can be thought of as having multiple landing sites within the vicinity and having the cargo advance toward other objectives after landing thus clearing the area for more cargo.

The edges between the different nodes represent the distance between the nodes and the assets that must carry the cargo between the nodes. The first set is between the AB and sea base. Along this path would be cargo carrying ships such as RO-RO, container ships, heavy lift aircraft, or a new design HSC. The second set is between the sea base and the objective. Assets traveling this edge could be airlift, such as CH-53 Sea Stallions, MV-22 Ospreys, etc., or sealift, such as LCAC, Utility Landing Craft (LCU), or a HLSL. A third set of edges could connect the AB with the objective; however, landing craft do not typically have the range to extend to the AB nor do cargo ships have the ability to land on the beach. Potentially, aircraft such as the MV-22 could be used on this theoretical edge, but this possibility was not included in the model.

The interface between the nodes and edges is handled in terms of loading and unloading times as well as cargo transfer sites. The load and unload times occur at all nodes, but the cargo transfer sites only exist at the ITS node. The ITS has a finite number of air, side and well deck transfer sites. Each cargo and lift asset, when transferring cargo occupies a certain number of these sites depending on the site type and the size of the asset.

The cargo to be transported represents the MEB as it is intended to be deployed. Units are grouped by sorties that should reach the objective at about the same time. For example, one sortie may be a tank platoon. The cargo is sized using three components, weight (in tons), area (in square feet), and personnel. Each sortie is given a priority for landing and an objective to land (either by air or shore). The sortie is allowed to be divided into smaller segments in order to be transported to the objective. For instance a tank platoon of four tanks may be divided into four equal parts so that it may be transported by a wider range of lift assets. Finally, cargo transported via cargo ship to the ITS must be prepared before being sent to the shore. Thus a time penalty is applied to this cargo aboard the ITS before it is made available for loading onto lift assets.

A final aspect of the model is sustainment. Once at an objective, the combat force is assumed to require sustainment supplies from the ITS. The sustainment requirement is assumed to be in proportion to the weight of the cargo. All sustainment is delivered via air assets since the force requiring the sustainment is likely no longer on the beach.

Simulation Inputs

For design purposes, the simulation requires several inputs. These inputs can be divided into variables which represent the design space of the SoS and parameters which represent assumptions or uncertainties. The variables and parameters are listed below. Except where noted, no ranges are given for either variables or parameters since the simulation is unconstrained thus any set of positive real inputs is possible though some inputs may result in extremely long delivery times. To limit the scope of the design space, it is assumed that only three ships are to be designed, a HSC, an ITS, and a HLSL. All other ships, such as the ESG, LCAC, MV-22, etc. are considered fixed in their characteristics.

HSC Variables

- Number of HSC
- Cargo capacity of HSC
- Speed of HSC
- Load time of HSC
- Unload time of HSC
- Required transfer sites on ITS

ITS Variables

- Number of ITS (>0)
- Cargo capacity of ITS
- Number of transfer sites on ITS (>0)
- Sea and air lift assets carried by ITS

HLSL Variables

- Cargo capacity of HLSL
- Loaded speed of HLSL
- Unloaded speed of HLSL
- Load time of HLSL
- Unload time of HLSL
- Required transfer sites on ITS

Parameters

- Cargo list
- Distance between nodes
- Number and makeup of ESGs
- Number and makeup of non-HSC cargo ships
- Number and makeup of non-HLSL lift assets
- Support element on ITS
- Required sustainment of MEB
- Sustainment carried on ITS
- Number of daily flight hours
- Attrition rate
- Mechanical breakdown rate

Simulation Outputs

The outputs of the simulation are meant to measure the effectiveness of the sea base operating as a system. Thus three measures indicate the performance: the time to complete delivery of the cargo, and the usage efficiencies of cargo and lift assets. The usage efficiency is measured as the time spent completing the mission (loading, unloading, or transiting) over the total time to complete the delivery. Cargo and lift assets are assumed inefficient if they must wait to receive cargo because there is no cargo available to load or if there are no unoccupied transfer sites available. These three measures are by no means the only ones that measure the performance of the sea base, but they are quantitative, measurable, and understandable and can be useful for comparison purposes.

Simulation Operation

The simulation is conducted in discrete time steps. It begins with the cargo being allocated to the ITS, ESG, and available cargo ships. Though there are multiple ways to load the various ships, no optimization is performed by the performance model. Instead, the cargo elements aboard the ESG are specifically designated and are assumed to fit aboard the available ESG assets. The remaining cargo is preloaded onto the ITS based on the desired landing order, with cargo landing first being preloaded first. When the ITS reaches its capacity, the remaining cargo is loaded onto available cargo ships. Half the cargo ships are assumed to be in transit to the sea base to represent the supply train. All remaining cargo is assumed to be at the advance base. For simplicity, cargo is not assigned to individual ITS or ESG ships within the sea base. Thus, for the purpose of this simulation, the cargo on the ITS could be located on any of the ITS ships. The same goes for cargo on the ESG.

Cargo identified as being on the ESG or ITS is preloaded onto the various lift assets based on the priority of the cargo and the cargo capacity of the lift asset. Each lift asset is loaded with a fraction of a cargo sortie until it reaches its capacity.

The simulation begins at H-hour, where the lift assets land simultaneously at their respective objectives. After this point, each asset is assigned a mission (loading, unloading, transiting) and a time remaining to complete the mission. At each time interval, the time remaining is reduced and checked to see if the mission is complete. If complete, the mission is changed and a new time remaining is established.

The lift and cargo assets are handled slightly differently. Each lift asset unloads in its allotted time and then returns to the sea base at its unloaded speed. When the lift asset reaches the sea base, if it originally came from the ESG, it returns there to receive any remaining cargo still required to reach the objectives, otherwise it goes to the ITS. The lift assets going to the ITS queue up for an available transfer site(s). Similar to the treatment of cargo, for simplicity the ITS transfer sites are not handled by individual ships but rather as a total available among the ITS ships. Once a site becomes available, the lift asset that has been waiting the longest takes the site(s) and loads cargo for the next wave. Once the load time has elapsed, the lift asset proceeds to the objective at its loaded speed. Refueling of lift assets is assumed to occur in parallel with loading operations aboard either the ESG or ITS.

Cargo assets operate in much the same way except they are given priority for unloading cargo onto the ITS over the lift assets and only transfer cargo to the ITS. Once cargo has completed unloading, the cargo asset returns to the AB. The cargo, however, must wait a predetermined amount of time before being available to load onto lift assets.

Sustainment required by the cargo at the objective is updated at each time interval and is treated as a separate cargo sortie. This sortie is given the highest priority to transfer to the objective to ensure that the landing force is properly supplied. The ITS carries some amount of sustainment supplies on board which is drawn upon to send to the objectives. When a day's supply is consumed, new supplies are prepared at the AB to replenish the ITS via cargo ships.

Simulation Validation

There is limited information available on sea basing operational times, therefore the validation was performed in segments where data was available.

To validate the transportation of cargo from the sea base to the objective for a distance of 25nm, a special architecture of the model was used such that the sea base would have 55 loads of a standard LCAC. The number of LCACs available to transport the load and the number of interface sites at the sea base were varied. The results were compared with data obtained from two different logistics models developed by the Naval Sea Systems Command (NAVSEA) and the U.S. Marine Corps Combat Development Command (MCCDC). Figure G2 shows the comparisons graphically.

The mean error is 5.5% and the median error is 4.9%. The most severe errors occur with large number of LCAC and is due to lack of any lane confliction at the beach in the MIT Model. The other models assume only 20 LCAC can reach the beach at any one time. Looking only at simulations with less than 20 LCACs, the error drops to 2.7%.

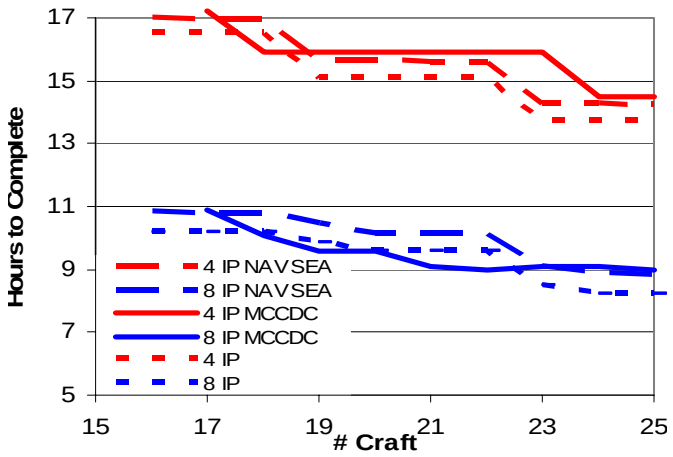


Figure G2. Sea Base to Objective Transport Validation For a Distance of 25nm

The second validation is used to check both the makeup of the MEB loadout as well as the MIT Model's ability to configure the loads of the ships. The validation is done against the scenario used in the U.S. Marine Corps Draft STOM CONOPS. In that scenario, 2 ESGs (1 LPD-17, 1 LSD-41, 1 LHA(R)) and a squadron of 6 MPF(F) ships with 10 organic LCU(R) and 28 aircraft operating spots carry the MEB. Of the 6 MPF(F) ships, 2 were assumed to be used for tactical aircraft and thus were not included in the logistics validation thus only 20 aircraft spots were available. Also, only 3 of the MPF(F) ships had two interfaces each for LCUs or LCAC lighters. For simplicity, the MPF(F) were combined into a single sea base platform, the ITS, with 20 aircraft and 6 surface interface spots.

The cargo lift requirements consist of the personnel and equipment for two vertical and two surface battalion landing teams allocated between the ESG and the MPF(F). Since it is assumed that the entire force would be carried aboard the sea base at the start of the conflict, the capacity of the ITS was made sufficiently large so as not to constrain the problem.

The LCAC and LCU(R) load times was approximated based on time given in the STOM CONOPS assault plan. Unload times were unable to be determined similarly so were assumed to be the same as load times. The results of the validation are shown in Table G1.

Table G1. Cargo Loading and Delivery Validation Results

	STOM CONOPS	MIT Sea Base Model	Error
LCAC Sorties	30	28	- 2
LCU(R) Sorties	18	18	0
MV-22 Sorties	195	196	+ 1
CH-53 Sorties	76	66	- 10
Air BLT Delivery Time (1^{st} to last landing)	7.27 hrs	7.01 hrs	-3.6%
Surface BLT Delivery Time (1^{st} to last landing)	6.81 hrs	5.78 hrs	15.1%

The results between the two models are very similar except for the surface Battalion Landing Teams (BLT) delivery time and aircraft sorties. The cause of the surface BLT deliver time discrepancy is two fold. First the STOM CONOPS assumes that the platforms in the ESG

were only serviced by their organic lighterage rather than being based on lift priority and available assets. More specifically, in the STOM CONOPS, the cargo aboard the LSD awaited the return of its LCUs while there were available LCACs that could have carried the load faster. The second cause of the discrepancy is the lack of interface conflict at the beach. While this is an issue at the initial surge, it should equalize out over time. As shown in the previous model validation, this can cause a couple of percentage points of error in the delivery time. The cause of the aircraft sorties appears to be differences in allocation of loads. While each aircraft is only allowed to carry equipment and personnel from a single sortie, the determination of whether a CH-53 or MV-22 is used can be different between the two models. In order to keep as much flexibility, the cargo was not specifically geared to be carried as a certain combination, but rather simply constrained by the aircraft's carrying capacity.

One final common error is the smearing effect. The sea base transfer node is made up of several ships, however, the MIT model does not specifically track each platform at the node but rather treats them as two units, an ESG and an ITS. The cargo contained aboard each and the interface spots are combined together. This can create some problems especially if some platforms carry significantly more cargo than others. This problem could be minimized through prior planning before the assault.

Appendix H: Performance Simulation Code

```
function [Output]=MissionSim5(CargoShips,JOAShips,Lighter,CargoLoads,Param)
%
% Mission Simulation Model
%
% Assumptions (In addition to ship characteristics)
%   2 objectives (one for surface forces, one for vertical forces)
%   All lighters deliver cargo simultaneously to objective at H-hour
%   All lighters are preloaded for initial assault (no load time required)
%   AAAV all deliver themselves and Cargo at H-hour
%   Sustainment required in linear proportion to cargo weight
%   Sustainment not required until delivered to the objective
%   Maintenance/Repair/Fueling included in load time
%   Aviation only active during certain hours starting at H-Hour
%   ITS lift sites and warehouse capability interchangable
%    (i.e. not broken down by individual ships)
%   No Site confliction for ESG platform lighters
%    (i.e. ESG platforms have sites for all their organic assets)
%   All ITS and ESG platforms are present in JOA at beginning of simulation
%   Three types of sites [Air, Side, Well Deck]
%   CargoLoadList identifies cargo preloaded on ESG
%   Non ESG cargo is placed on ITS, the Cargo Ships, then Advanced Base
%   Cargo is loaded in priority order from CargoLoadList
%   Cargo can be broken into equal # of pieces defined by fraction column
%
% User Inputs
% The simulation runs based on 5 inputs variables
%
%   CargoShips  .Name           % Name of ship
%               .Speed          % Transit & Return Speed (in kts)
%               .Range          % Range (not used)
%               .Number         % Number used by sea base
%               .Capacity       % Cargo Carrying Capability
%               .LoadTime       % Load Time (in hours)
%               .UnloadTime     % Unload Time (in hours)
%               .Sites          % Interface sites required
%   JOAShips    .Name           % Name of JOA ship
%               .Speed          % Speed (not used)
%               .Range          % Range (not used)
%               .Number         % Number at sea base
%               .Capacity       % Cargo Carrying Capability
%               .Sites          % Interface sites available
%               .Lighter        % Organic lighters on board
%               .AAAV           % Number of AAAVs on board
%   Lighter     .Name           % Name of lighter
%               .LSpeed         % Speed when loaded (in kts)
%               .USpeed         % Speed when unloaded (in kts)
%               .Capacity       % Cargo Carrying Capability
%               .LoadTime       % Loading time (in hours)
%               .UnloadTime     % Unloading time (in hours)
%               .Air            % Air or surface craft
%               .Sites          % Interface sites required
%               .Attrition      % Percent Chance of attrition per sortie
%     * .Capacity variable sizes based on Param.CargoSize
%       .Sites variable sizes based on Param.SiteSize
```

```
%       .Lighter variable size based on # of element of Lighter variable
%
%
%   CargoLoads (Each row is one sortie of the joint force)
%               Column 1        % Size in tons
%               Column 2        % Size in sq. ft. footprint
%               Column 3        % Size in personnel
%               Column 4        % Initial Location (0-AB, 1-ITS, 2-ESG)
%               Column 5        % Number of parts the sortie can be divided
%               Column 6        % Desired landing time / Landing priority
%               Column 7        % Which objective (1-air, 2-shore)
%   Param       .NumITS         % Number of ITS Variants listed before ESG
%               .CargoSize      % Number of Capacity Variables
%               .SiteSize       % Number of interface sites
%               .TimeSteps      % # time steps / hr (12 = 5 min per step)
%               .Distance       % [SB to Surf Obj, SB to Air Obj, AB to SB]
%               .DailySustain   % Required sustainment of CargoList per day
%               .DaysSustain    % # Days Sustainment on ITS at beginning
%               .AAAVSize       % Size of AAAV (in Cargo Capacity terms)
%               .AAAVSpeed      % Speed of AAAV (in kts)
%               .SupportElement % Support Element (reduces ITS capacity)
%               .FlightHours    % Hrs per day air lighterage allowed to fly
%               .PrepTime       % Hrs prep required on ITS
%               .DayMax         % Max days until simulation termination
%               .PlotFreq       % Plotting Frequency (in Time Steps)
%               .PlotFlag       % 0 - No plots
%
% Outputs
%   Output      .DeliveryTime   % Time to complete delivery of cargo list
%               .LiftEff        % Percent Lift Asset Utilization
%               .CargoEff       % Percent Cargo Asset Utilization
%               .LiftErr        % Error between desired and actual delivery
%               .MaxWarehouse   % Maximum warehouse capacity used by ITS
%               .CargoUsage     % Number of cargo assets actually used
%
%
% Created by Robert Wolf - Aug 15, 2004
%
%*****************************************
% Convert Param Structure Variable to normal variables
%*****************************************
NumITS=Param.NumITS;
CargoSize=Param.CargoSize;
PlotFreq=Param.PlotFreq;
PlotFlag=Param.PlotFlag;
TimeSteps=Param.TimeSteps;
Distance=Param.Distance;
DailySustain=Param.DailySustain;
DaysSustain=Param.DaysSustain;
AAAVSize=Param.AAAVSize;
AAAVSpeed=Param.AAAVSpeed;
SupportElement=Param.SupportElement;
FlightHours=Param.FlightHours;
PrepTime=Param.PrepTime;
%
Time=0;Day=0;Hour=0;Min=0;
TotalLiftWait=0;TotalLift=0;
```

```
TotalCargo=0;TotalCargoWait=0;
LiftErr=0;
%*****************************************
% Temporary Validation Variable
%*****************************************
Output.CargoResults=zeros(1,4);
%
%*****************************************
% Setup and Simulation Code starts here
%*****************************************
%
% Setup Cargo Tracking Variables
NullCargo=zeros(1,CargoSize);
NullSites=zeros(1,Param.SiteSize);
AtBase=NullCargo;
Cargo=NullCargo;
Warehouse=NullCargo;
ESGWarehouse=NullCargo;
OnLighter=NullCargo;
Delivered=NullCargo;
Prep=zeros(PrepTime+1,CargoSize);
PrepList(PrepTime).List=[];
Output.BLTDeliveryTime=[0 0];
%
% Get Data on the Ships and Cargo to be used
%[CargoShips,JOAShips,Lighter]=GetShipData2;
%CargoLoads=CargoLoadList2;
%
% Establish Cargo and Sustainment Variables
JOASustain=sum(DaysSustain.*DailySustain);
CargoSustain=NullCargo;
%
% AAAVs handled seperately from Cargo list since require no lighterage
%   They are included only to ensure the warehouse requirements are met
ITSAAAV=sum([JOAShips(1:NumITS).AAAV]);
ESGAAAV=sum([JOAShips(NumITS+1:length(JOAShips)).AAAV]);
AAAVTransit=ceil(Distance(1)/AAAVSpeed*TimeSteps);
%
% ITS Warehouse includes Sustainment, Support Element, AAAV, and Cargo
Warehouse=Warehouse+JOASustain;
Warehouse=Warehouse+SupportElement;
Warehouse=Warehouse+ITSAAAV.*AAAVSize;
%
% ESG Warehouse made up of AAAV and MEB Deployment preloaded on ESG
ESGWarehouse=ESGWarehouse+ESGAAAV.*AAAVSize;
%
% MEB made up of the MEB Deployment preloaded and at AB and AAAVs
MEB=sum(CargoLoads(:,1:CargoSize));
MEB=MEB+(ITSAAAV+ESGAAAV).*AAAVSize;
%
% Setup the ITS Squadron data (i.e. open sites and cargo onboard)
WarehouseMax=[JOAShips(1:NumITS).Number]*reshape([JOAShips(1:NumITS).Capacity
],CargoSize,NumITS)';
SitesOpen=[JOAShips(1:NumITS).Number]*reshape([JOAShips(1:NumITS).Sites],Para
m.SiteSize,NumITS)';
ESGSitesOpen=sum([JOAShips(NumITS+1:length(JOAShips)).Number]); % ESG Sites
(Assumes 1 site per ship)
```

```
ESGSitesMax=ESGSitesOpen;
%
% Setup the Cargo List (CargoList(1) is Sustainment to be set up later)
%  CargoList is in order of the priority established in input file
[CargoLoadList,CargoIndex]=sortrows(CargoLoads,6);% Sort based on Priority
for i=1:1:size(CargoLoadList,1)   % First on CargoList is Sustainment
  CargoList(i+1).Location=CargoLoadList(i,4);  % 0-AB,1-ITS,2-ESG,3-Other
  CargoList(i+1).Number=0;                     % Ship Number
  CargoList(i+1).Priority=CargoLoadList(i,6);  % Desired Landing Priority
  CargoList(i+1).Air=CargoLoadList(i,7);       % Objective
  CargoList(i+1).Total=[CargoLoadList(i,1:3)]; % Total Cargo in Sortie
  CargoList(i+1).Fraction=CargoLoadList(i,5);  % # parts sortie divisible
  CargoList(i+1).InJOA=0;                      % Track whats on ITS/ESG
  CargoList(i+1).OnBeach=0;                    % Track whats at objective
  CargoList(i+1).Destroyed=0;
end
% Set up the Sustainment CargoList
CargoList(1).Location=1;              % Sustainment assumed all on ITS
CargoList(1).Total=sum(DailySustain); % Daily Sustainment reqmt
CargoList(1).Number=0;                % Don't particularly care
CargoList(1).Priority=0;              % Max priority
CargoList(1).Air=1;                   % Assume all sustainment airlifted
% Set the sustainment fraction so all lighters can transport
%  (Assume tons most limiting)
CargoList(1).Fraction=sum(DailySustain(:,1))/min([Lighter.Capacity]);
% InJOA is increased each time step.
%   If >0, one fraction will be sent to objective
CargoList(1).InJOA=0;                   % Amount required
CargoList(1).OnBeach=0;                 % Amount on Beach
%
% Allocate the Equipment to the various locations
%   (Note: ESG equipment should already be allocated)
%   Priority is ITS, Cargo Ships, Advance Base
%
% Allocate the Equipment aboard the ITS Ships
%   Also count the cargo on ESGs
for i=2:1:size(CargoList,2) % Skip Sustainment
  switch CargoList(i).Location
    case 0  % On Advanced Base
      if CargoList(i).Total<=WarehouseMax-Warehouse
        % If it is within the capacity of the ITS Warehouse, then load it
        CargoList(i).Location=1; % Specify it is on the ITS
        Warehouse=Warehouse+CargoList(i).Total; % Update cargo tracking
        CargoList(i).InJOA=CargoList(i).Fraction; % Update cargo tracking
        % InJOA defines # of pieces remaining to be sent to the objective
      else
        AtBase=AtBase+CargoList(i).Total;
      end
    case 1  % On ITS
      Warehouse=Warehouse+CargoList(i).Total; % see notes for case 1
      CargoList(i).InJOA=CargoList(i).Fraction;
    case 2  % On ESG
      ESGWarehouse=ESGWarehouse+CargoList(i).Total; % see notes for case 2
      CargoList(i).InJOA=CargoList(i).Fraction;
    case 3  % On Cargo Ships
      % Shouldn't happen but may want to be able to preload (i.e. MPS)
  end
```

```
end
%
% Setup the Cargo Ships in Operation and preload cargo from Advanced base
%
% Each Cargo Asset assigned to a structure variable [CargoAsset(i)]
%
count=1;
for i=1:size(CargoShips,2)
  % Determine the spacing of the incoming cargo ships by taking the time
  % from start of loading to getting to the JOA and assuming 1/2 of the
  % cargo assets are in that process. The first asset will arrive after a
  % delay so as not to interfere with the assault.

TurnaroundTime=(Distance(3)/CargoShips(i).Speed+CargoShips(i).LoadTime)*TimeS
teps;
  % Assume each cargo ship is loaded and ready
  CargoSpacing=round(1.1*CargoShips(i).LoadTime*TimeSteps);
  for j=1:CargoShips(i).Number
    % For all cargo assets, define certain default values
    CargoAsset(count).Type=i;
    CargoAsset(count).Mission='Transit';
    CargoAsset(count).TimeRem=CargoSpacing*j;
    CargoAsset(count).Air=Lighter(i).Air;  % 1 - Airlift, 0 - Sealift
    CargoAsset(count).Sites=NullSites;     % number of sites occupied
    CargoAsset(count).Sortie=0;            % number of sorties completed
    CargoAsset(count).Capacity=CargoShips(i).Capacity;
    CargoAsset(count).Cargo=NullCargo;
    CargoAsset(count).CargoList=[];
    count=count+1;
  end
end
%
% Now preload the cargo aboard the Cargo Ships
for j=1:1:size(CargoAsset,2)
  for i=2:1:size(CargoList,2) % Skip Sustainment
    if ~CargoList(i).Location % Is it at Advanced Base (Location=0)
      if CargoList(i).Total<=CargoAsset(j).Capacity-CargoAsset(j).Cargo
        % If within the remaining capacity of cargo ship, then load it
        CargoList(i).Location=3;
        CargoList(i).Number=j;
        CargoAsset(j).Cargo=CargoAsset(j).Cargo+CargoList(i).Total;
        CargoAsset(j).CargoList=[CargoAsset(j).CargoList, i];
        Cargo=Cargo+CargoList(i).Total;
        AtBase=AtBase-CargoList(i).Total;
      end
    end
  end
  if all(~CargoAsset(j).Cargo)
    % If no cargo was loaded, then have cargo ship wait at advance base
    CargoAsset(j).Mission='Loading';
    CargoAsset(j).TimeRem=1;
  end
end
%
% The transit time is calculated as distance/speed.  Return and Transit
% times for each platform used later
Transit=((Distance(2)*[Lighter.Air])./[Lighter.LSpeed]+...
```

```
  (Distance(1)*~[Lighter.Air])./[Lighter.LSpeed]).*TimeSteps;
Return=((Distance(2)*[Lighter.Air])./[Lighter.USpeed]+...
  (Distance(1)*~[Lighter.Air])./[Lighter.USpeed]).*TimeSteps;
Delay=(max(Transit)-Transit);
%
% Setup the Lighterage Ships in Operation
%
% Each Lift Asset (air and surface) assigned to a structure variable
%   [LiftAsset(i,j)] where i identifies the lift platform type
%
count=1;
for i=1:size(Lighter,2)
  for j=1:size(JOAShips,2)
    for k=1:JOAShips(j).Lighter(i)
      for l=1:JOAShips(j).Number
        % For all lift assets, define certain default values
        LiftAsset(count).Type=i;
        LiftAsset(count).Origin=j; % ITS/ESG platform that it belongs
        LiftAsset(count).Air=Lighter(i).Air;  % 1 - Airlift, 0 - Sealift
        LiftAsset(count).Sites=NullSites; % No sites occupies at start
        LiftAsset(count).Sortie=0;        % The number of sorties completed
        LiftAsset(count).Capacity=Lighter(i).Capacity;
        LiftAsset(count).Cargo=NullCargo;
        LiftAsset(count).CargoList=[];
        LiftAsset(count).Mission='Transit'; % Everything starts preloaded
        LiftAsset(count).TimeRem=1;         % Everything arrives at H-hour
        % Find which cargos are available on the right platform and going
        % to the right objective
        a=[LiftAsset(count).Air]==[CargoList.Air]&[CargoList.InJOA]>0&...
          or(and(j,[CargoList.Location]==1),and(j,[CargoList.Location]==2));
        % PreLoad the Cargo
        % Load the lighter with as much cargo as it can carry
        %  ***Note*** this may not be the exact same loadout at STOM CONOPS
        for m=2:1:length(a) % Skip Sustainment
          if a(m)
            % Size of cargo segment
            CargoFraction=CargoList(m).Total./CargoList(m).Fraction;
            CapacityLeft=LiftAsset(count).Capacity-LiftAsset(count).Cargo;
            if LiftAsset(count).Air==1&any(LiftAsset(count).Cargo)
              CapacityLeft=NullCargo;
            end %(Air should only carry one sortie)
            while all(CapacityLeft>=CargoFraction)&... % Capacity avail?
                CargoList(m).InJOA>0 % Cargo left to load?
              % Load the cargo on the lift asset, remove it from the
              % warehouse, and keep track of which cargo you are carrying
              % (LiftAsset.CargoList)
              LiftAsset(count).Cargo=LiftAsset(count).Cargo+CargoFraction;
              CapacityLeft=CapacityLeft-CargoFraction;
              CargoList(m).InJOA=CargoList(m).InJOA-1;
              LiftAsset(count).CargoList=[LiftAsset(count).CargoList, m];
              OnLighter=OnLighter+CargoFraction;
              if CargoList(m).Location==1
                Warehouse=Warehouse-CargoFraction;
              elseif CargoList(m).Location==2
                ESGWarehouse=ESGWarehouse-CargoFraction;
              end
            end
```

```
          end
        end
        LiftAsset(count).Sortie=LiftAsset(count).Sortie+1;
        count=count+1;
      end
    end
  end
end
%
%
% Commence Simulation Time
MEBToGo=sum([CargoList.Fraction])-CargoList(1).Fraction-...
  sum([CargoList.OnBeach])+CargoList(1).OnBeach;
MaxWarehouse=Warehouse;
while and(MEBToGo>0,Day<Param.DayMax)
  %
  % Determine the DD:HH:MM time
  Min=rem(fix(Time*60/TimeSteps),60);
  Hourold=Hour;
  Hour=rem(fix(Time/TimeSteps),24);
  Day=fix(Time/TimeSteps/24);
  %
  % If we have switched to a new hour move equipment preparations along
  if Hourold~=Hour
    if ~isempty([PrepList(1).List]) % Equipment completing preparations?
      % Identify it as ready for loading onto lighter (Location=1,InJOA>0)
      for i=1:1:size(PrepList(1).List,2)
        CargoList(PrepList(1).List(1,i)).Location=1;
        CargoList(PrepList(1).List(1,i)).InJOA=...
          CargoList(PrepList(1).List(1,i)).Fraction;
      end
    end
    % Advance preparation by one hour and clear last hour
    for i=2:1:PrepTime
      PrepList(i-1).List=PrepList(i).List;
    end
    Prep(1:PrepTime-1,:)=Prep(2:PrepTime,:);
    Prep(PrepTime,:)=NullCargo;
    PrepList(PrepTime).List=[];
  end
  %
  % Take care of the AAAV (if required)
  if Time==1  % Time Step #1 - Launch AAAV
    Warehouse=Warehouse-ITSAAAV.*AAAVSize;
    ESGWarehouse=ESGWarehouse-ESGAAAV.*AAAVSize;
    %         OnLighter=OnLighter+(ITSAAAV+ESGAAAV).*AAAVSize;
    %     end
    %     if Time==max(Transit) % AAAV have hit the beach
    %         OnLighter=OnLighter-(ITSAAAV+ESGAAAV).*AAAVSize;
    Delivered=Delivered+(ITSAAAV+ESGAAAV).*AAAVSize;
  end
  %
  % Go through each Cargo Asset to check mission status
  %   (Cargo Assets done first to ensure priority)
  for i=1:size(CargoAsset,2)
    % Reduce the time remaining to the next mission
    CargoAsset(i).TimeRem=CargoAsset(i).TimeRem-1;
```

```
    if CargoAsset(i).TimeRem<=0  % If the mission has been completed then
      switch CargoAsset(i).Mission
        % For each mission except waiting,
        %   Define new mission
        %   Reset Time to complete new mission
        %   Reset the Sites and Start Prep Time for Cargo
        case 'Unloading'
          CargoAsset(i).Mission='Returning';
          CargoAsset(i).TimeRem=(Distance(3)/...
            CargoShips(CargoAsset(i).Type).Speed)*TimeSteps;
          SitesOpen=SitesOpen+CargoAsset(i).Sites;
          CargoAsset(i).Sites=NullSites;
          Prep(PrepTime,:)=Prep(PrepTime,:)+CargoAsset(i).Cargo;
          Prep(PrepTime+1,:)=Prep(PrepTime+1,:)-CargoAsset(i).Cargo;
          PrepList(PrepTime).List=...
            [PrepList(PrepTime).List, CargoAsset(i).CargoList];
          CargoAsset(i).Cargo=NullCargo;
          CargoAsset(i).CargoList=[];
          CargoAsset(i).Sortie=CargoAsset(i).Sortie+1;
        case 'Returning'
          CargoAsset(i).Mission='Loading';
          CargoAsset(i).TimeRem=...
            CargoShips(CargoAsset(i).Type).LoadTime*TimeSteps;
        case 'Loading'
          % Check and see if there is a need for sustainment at ITS
          %  Order 1 day of sustainment if >1/2 day used and not ordered
          if any(round((JOASustain(1:2)+CargoSustain(1:2))./...
              sum(DailySustain(:,1:2)))-DaysSustain)
            CargoAsset(i).Cargo=sum(DailySustain);
            CargoAsset(i).CargoList=[1]; % Should not have any other cargo
            CargoSustain=CargoSustain+sum(DailySustain); % Track order
          end
          a=~[CargoList.Location]; % List of only cargo on advanced base
          if any(a)   % Any cargo waiting at advanced base?
            for j=1:1:size(a,2)
              if a(j) % Cargo at AB and able to fit on cargo ship?
                if CargoList(j).Total<=CargoAsset(i).Capacity-...
                    CargoAsset(i).Cargo
                  % If so, update cargo list, and track of cargo on ship
                  %  Note:  Entire item is carried aboard cargo ships.
                  %         No fractional amounts allowed
                  CargoList(j).Location=3;
                  CargoList(j).Number=i;  % Cargo Ship number
                  CargoAsset(i).Cargo=CargoAsset(i).Cargo+...
                    CargoList(j).Total;
                  CargoAsset(i).CargoList=[CargoAsset(i).CargoList, j];
                  Cargo=Cargo+CargoList(j).Total;
                end
              end
            end
          end
          if any(CargoAsset(i).Cargo)
            % If the ship is loaded with cargo then update mission and time
            CargoAsset(i).Mission='Transit';

CargoAsset(i).TimeRem=(Distance(3)/CargoShips(CargoAsset(i).Type).Speed)*Time
Steps;
```

```
        else
          % If ship has no cargo, leave it at AB until next time step
          CargoAsset(i).TimeRem=1;
        end
      case 'Transit'
        CargoAsset(i).Mission='Waiting';
        % No change to Time Remaining since priority based on negative
        % Time Remaining
      case 'Waiting'
        % Do nothing
      otherwise
        % Error Unknown Mission or maybe destroyed??
    end
  end
end
%
% Go through each Lift Asset and check mission status
for i=1:size(LiftAsset,2)
  LiftAsset(i).TimeRem=LiftAsset(i).TimeRem-1;
  if LiftAsset(i).TimeRem<=0  % If the mission has been completed then...
    switch LiftAsset(i).Mission
      % For each mission except waiting,
      %   Define new mission
      %   Reset Time to complete new mission
      %   Keep track of cargo transferred and lift sites used
      case 'Loading'          % For the loading mission
        if and(Hour>FlightHours,LiftAsset(i).Air)
          % Flight missions only during flighthours, so prevent going to
          % next mission keeps a site occupied, but does not hinder
          % efficiency
          LiftAsset(i).Mission='Crew Rest';
          LiftAsset(i).TimeRem=24*TimeSteps-mod(Time,24*TimeSteps)-1;
        else
          LiftAsset(i).Mission='Transit';
          Dist=Distance(LiftAsset(i).Air+1);
          LiftAsset(i).TimeRem=(Dist/...
            Lighter(LiftAsset(i).Type).LSpeed)*TimeSteps;
          if LiftAsset(i).ESGLoad&~LiftAsset(i).Air
            ESGSitesOpen=min(ESGSitesMax,ESGSitesOpen+1);
          else
            SitesOpen=SitesOpen+LiftAsset(i).Sites;
          end
          LiftAsset(i).Sites=NullSites;
          LiftAsset(i).Sortie=LiftAsset(i).Sortie+1;
        end
      case 'Transit'
        LiftAsset(i).Mission='Unloading';
        LiftAsset(i).TimeRem=...
          Lighter(LiftAsset(i).Type).UnloadTime*TimeSteps;
        if rand(1)<Lighter(LiftAsset(i).Type).Attrition
          LiftAsset(i).Mission='Attrition';
          LiftAsset(i).TimeRem=9999;
          OnLighter=OnLighter-LiftAsset(i).Cargo;
          Delivered=Delivered+LiftAsset(i).Cargo;
          for j=1:1:size(LiftAsset(i).CargoList,2)
            CargoList(LiftAsset(i).CargoList(j)).OnBeach=...
              CargoList(LiftAsset(i).CargoList(j)).OnBeach+1;
```

```
            CargoList(LiftAsset(i).CargoList(j)).Destroyed=...
              CargoList(LiftAsset(i).CargoList(j)).Destroyed+1;
            if LiftAsset(i).CargoList(j)==1 %Is it sustainment?
              Delivered=Delivered-CargoList(1).Total/CargoList(1).Fraction;
            end
          end
        end
      case 'Unloading'
        LiftAsset(i).Mission='Returning';
        Dist=Distance(LiftAsset(i).Air+1);
        LiftAsset(i).TimeRem=(Dist/...
          Lighter(LiftAsset(i).Type).USpeed)*TimeSteps;
        OnLighter=OnLighter-LiftAsset(i).Cargo;
        Delivered=Delivered+LiftAsset(i).Cargo;
        for j=1:1:size(LiftAsset(i).CargoList,2)
          CargoList(LiftAsset(i).CargoList(j)).OnBeach=...
            CargoList(LiftAsset(i).CargoList(j)).OnBeach+1;
          if LiftAsset(i).CargoList(j)==1 %Is it sustainment?
            Delivered=Delivered-CargoList(1).Total/CargoList(1).Fraction;
          elseif CargoList(LiftAsset(i).CargoList(j)).OnBeach==...
              CargoList(LiftAsset(i).CargoList(j)).Fraction
            LiftErr=LiftErr+Time*60/TimeSteps-...
              CargoList(LiftAsset(i).CargoList(j)).Priority;
          end
        end
        LiftAsset(i).Cargo=NullCargo;
        LiftAsset(i).CargoList=[];
        % Place a damage/maintenance check here
        %   If damaged, can unload cargo, but put time penalty into
        %   return time i.e. slower return speed and repair time
      case 'Returning'
        LiftAsset(i).Mission='Waiting';
        % No change to Time Remaining since priority based on time spent
waiting
      case 'Waiting'
        % Do nothing
      case 'Crew Rest'
        LiftAsset(i).Mission='Loading';
        LiftAsset(i).TimeRem=1;
      case 'Maintenance'
      case 'Attrition'
      otherwise
        % Error Unknown Mission
    end
  end
end
%
% Place cargo assets in open lift sites based on wait time
% (Cargo ships done first to give them priority in sites)
%   Ca is wait time for each cargo asset (should be NaN for empty fields)
%   Cindex indicates the position if all numbers placed in column vector
[Ca,Cindex]=sort([CargoAsset.TimeRem]);   % Waiting Cargo Ships are <=0
% Start with the cargo asset that has waited the longest
for k=1:size(Ca,2)
  i=CargoAsset(Cindex(k)).Type;  % Determine the type of platform
  if Ca(k)>0; break; end         % If no more waiting ships, then stop
  if all(~SitesOpen); break; end % If all sites taken, then stop
```

```
  % If no room for cargo then wait for next item
  if WarehouseMax-Warehouse<=CargoAsset(Cindex(k)).Cargo;break;end
  % for each waiting ship, check each site configuration to see if there
  % are enough open slots.  Priority order given based on wait time
  % Preferred site defined by site order in GetShipData
  for l=1:size(CargoShips(i).Sites,2)
    if CargoShips(i).Sites(l)<=SitesOpen(l)&CargoShips(i).Sites(l)~=0
      % If there are enough open slots then...
      % Change mission, load time, adjust open slots and update cargo
      CargoAsset(Cindex(k)).Mission='Unloading';
      CargoAsset(Cindex(k)).TimeRem=CargoShips(i).UnloadTime*TimeSteps;
      SitesOpen(l)=SitesOpen(l)-CargoShips(i).Sites(l);
      CargoAsset(Cindex(k)).Sites(l)=CargoShips(i).Sites(l);
      if CargoAsset(Cindex(k)).CargoList(1)==1
        % If part of the cargo was sustainment, then update sustainment
        % cargo tracking and remove sustainment from the cargo so that it
        % looks like a regular shipment
        JOASustain=JOASustain+sum(DailySustain);
        Warehouse=Warehouse+sum(DailySustain);
        CargoSustain=CargoSustain-sum(DailySustain);
        CargoAsset(Cindex(k)).Cargo=CargoAsset(Cindex(k)).Cargo-...
          sum(DailySustain);
        a=size(CargoAsset(Cindex(k)).CargoList,2);
        CargoAsset(Cindex(k)).CargoList(1:a-1)=...
          CargoAsset(Cindex(k)).CargoList(2:a);
      end
      Cargo=Cargo-CargoAsset(Cindex(k)).Cargo;
      Warehouse=Warehouse+CargoAsset(Cindex(k)).Cargo;
      Prep(PrepTime+1,:)=Prep(PrepTime+1,:)+CargoAsset(Cindex(k)).Cargo;
      break % cargo asset in cargo site, so stop checking sites
    end
  end
  % if unable to load because insufficient open slot, priority will
  % improve as time continues
end
%
% Place lift assets in open lift sites based on wait time
%   aaa is wait time for each lift asset (should be NaN for empty fields)
%   index is indicates the position before sort
[aaa,index]=sort([LiftAsset.TimeRem]);   % Waiting Lift Ships are <= 0
% Start with the lift asset that has waited the longest
for k=1:size(aaa,2)
  yy=LiftAsset(index(k)).Type;  % Determine the type of platform
  if aaa(k)>0; break; end       % If no more ships are waiting, then stop
  % If the Lift Asset originated from an ESG platform and there is ESG
  % Cargo left then get that cargo first
  if LiftAsset(index(k)).Origin>NumITS&any(round(ESGWarehouse))
    % Get list which shows any cargo going to same location and on ESG
    a=[CargoList.Location]==2&[CargoList.InJOA]>0&[CargoList.Air]==...
      LiftAsset(index(k)).Air;
    if any(a)&ESGSitesOpen % Anything there? and site available
      for i=1:1:size(a,2)
        if LiftAsset(index(k)).Air&any(LiftAsset(index(k)).Cargo);
          break;
        end
        if a(i) % Is this item available to load?
          CargoFraction=CargoList(i).Total./CargoList(i).Fraction;
```

```
        CapacityLeft=LiftAsset(index(k)).Capacity-...
          LiftAsset(index(k)).Cargo;
        % Load cargo until no more spare capacity or item fully loaded
        % Update cargo and keep track of which cargo is on board
        while all(CapacityLeft>=CargoFraction)&CargoList(i).InJOA>0
          LiftAsset(index(k)).Cargo=...
            LiftAsset(index(k)).Cargo+CargoFraction;
          CapacityLeft=CapacityLeft-CargoFraction;
          CargoList(i).InJOA=CargoList(i).InJOA-1;
          OnLighter=OnLighter+CargoFraction;
          ESGWarehouse=ESGWarehouse-CargoFraction;
          LiftAsset(index(k)).CargoList=...
            [LiftAsset(index(k)).CargoList, i];
        end
      end
    end
    if any(LiftAsset(index(k)).Cargo) % Was cargo loaded?
      LiftAsset(index(k)).Mission='Loading';
      Dist=Distance(LiftAsset(index(k)).Air+1);
      LiftAsset(index(k)).TimeRem=(Lighter(yy).LoadTime)*TimeSteps;
      LiftAsset(index(k)).ESGLoad=1;
      if ~LiftAsset(index(k)).Air
        ESGSitesOpen=ESGSitesOpen-1;
      end
    end
  end
end
% If not an ESG lighter or no ESG cargo (i.e. the Asset is still
% waiting) for each waiting ship, check each site configuration to see
% if there are enough open slots
% Preferential site based on order of site configuration in GetShipData
if LiftAsset(index(k)).Mission=='Waiting'
  if any(SitesOpen>=Lighter(yy).Sites) % Enough potential sites open?
    for l=1:size(Lighter(yy).Sites,2)
      if Lighter(yy).Sites(l)<=SitesOpen(l)&Lighter(yy).Sites(l)~=0
        % If there are enough open slots then...
        % Check to see if cargo available, otherwise continue waiting
        %   This ensures an open site for incoming cargo, otherwise may
        %   never get resupply
        %
        % Get array which shows cargo going to same location and on ITS
        a=[CargoList.Location]==1&[CargoList.InJOA]>0&...
          [CargoList.Air]==LiftAsset(index(k)).Air;
        if any(a)   % Any cargo available
          for i=1:1:size(a,2)
            if LiftAsset(index(k)).Air&any(LiftAsset(index(k)).Cargo);
              break;
            end
            if a(i) % Specific item available?
              CargoFraction=CargoList(i).Total./CargoList(i).Fraction;
              CapacityLeft=LiftAsset(index(k)).Capacity-...
                LiftAsset(index(k)).Cargo;
              % Load cargo until no more capacity or item fully loaded
              % Update cargo and keep track of which cargo is on board
              while all(CapacityLeft>=CargoFraction)&CargoList(i).InJOA>0
                LiftAsset(index(k)).Cargo=LiftAsset(index(k)).Cargo+...
                  CargoFraction;
```

```
                CapacityLeft=CapacityLeft-CargoFraction;
                CargoList(i).InJOA=CargoList(i).InJOA-1;
                OnLighter=OnLighter+CargoFraction;
                Warehouse=Warehouse-CargoFraction;
                LiftAsset(index(k)).CargoList=...
                  [LiftAsset(index(k)).CargoList, i];
                if i==1 % Carrying Sustainment?
                  JOASustain=JOASustain-CargoFraction;
                end
              end
            end
          end
          if any(LiftAsset(index(k)).Cargo~=NullCargo)
            % If Cargo loaded, then change mission, loadtime, open sites
            LiftAsset(index(k)).Mission='Loading';
            LiftAsset(index(k)).TimeRem=Lighter(yy).LoadTime*TimeSteps;
            SitesOpen(l)=SitesOpen(l)-Lighter(yy).Sites(l);
            LiftAsset(index(k)).Sites(l)=Lighter(yy).Sites(l);
            LiftAsset(index(k)).ESGLoad=0;
            break % lift asset in lift site, so stop checking
          else
            break % no Cargo even though site available, so stop
          end
        end
      end
    end
  end
end
%
% Account for use of sustainment supplies
CargoList(1).InJOA=CargoList(1).InJOA+Delivered(1)/MEB(1)/24/...
  TimeSteps*CargoList(1).Fraction;
%
Time=Time+1; % increment time step
MEBToGo=round(sum([CargoList.Fraction])-CargoList(1).Fraction-...
  sum([CargoList.OnBeach])+CargoList(1).OnBeach);
if ~dot([CargoList(2:length(CargoList)).Fraction]-...
    [CargoList(2:length(CargoList)).OnBeach],...
    [CargoList(2:length(CargoList)).Air]==0)&~Output.BLTDeliveryTime(1)
  Output.BLTDeliveryTime(1)=Time/TimeSteps;
end
if ~dot([CargoList(2:length(CargoList)).Fraction]-...
    [CargoList(2:length(CargoList)).OnBeach],...
    [CargoList(2:length(CargoList)).Air]==1)&~Output.BLTDeliveryTime(2)
  Output.BLTDeliveryTime(2)=Time/TimeSteps;
end
%
% Stuff to condense the mission status into a history (one character per
% lift asset)and for plotting the status
TotalLiftWait=TotalLiftWait+sum(strcmp({LiftAsset.Mission},'Waiting'));
TotalLift=TotalLift+size(LiftAsset,2);
TotalCargoWait=TotalCargoWait+sum(strcmp({CargoAsset.Mission},'Waiting'));
TotalCargo=TotalCargo+size(CargoAsset,2);
MaxWarehouse=max([MaxWarehouse;Warehouse]);
%
% Instructions to plot the mission status of the lift assets
```

```
if PlotFlag&or(~mod(Time-1,PlotFreq),~MEBToGo) % Plot at PlotFreq and end
  l=size(LiftAsset,2)+1;
  k=l+size(CargoAsset,2)-1;
  for i=1:1:size(Lighter,2)
    plotdisplay(i,1:l-1)=[strcmp({LiftAsset.Mission},'Loading')+...
      strcmp({LiftAsset.Mission},'Transit').*2+...
      strcmp({LiftAsset.Mission},'Unloading').*3+...
      strcmp({LiftAsset.Mission},'Returning').*4+...
      strcmp({LiftAsset.Mission},'Waiting').*5+...
      strcmp({LiftAsset.Mission},'Crew Rest').*6+...
      strcmp({LiftAsset.Mission},'Attrition').*7].*...
      ([LiftAsset.Type]==i);
  end
  for j=1:1:size(CargoShips,2)
    plotdisplay(j+size(Lighter,2),l:k)=[...
      strcmp({CargoAsset.Mission},'Loading')+...
      strcmp({CargoAsset.Mission},'Transit').*2+...
      strcmp({CargoAsset.Mission},'Unloading').*3+...
      strcmp({CargoAsset.Mission},'Returning').*4+...
      strcmp({CargoAsset.Mission},'Waiting').*5].*...
      ([CargoAsset.Type]==j);
  end
  figure(1)
  subplot(2,1,1)
  plot([1:1:size(plotdisplay,2)],plotdisplay,'.')
  char(sum(plotdisplay)+48);
  legend(char(Lighter.Name,CargoShips.Name),-1)
  axis([1 size(plotdisplay,2) 1 7])
  set(gca,'YTick',[1 2 3 4 5 6 7])
  set(gca,'XTick',[])
  set(gca,'YTickLabel',{'Loading','Transit','Unloading','Returning',...
    'Waiting','Crew Rest','Attrition'})
  Timestr=Hour*100+Min;
  title(['Time: D' num2str(Day,'+%.0f') ':' ...
    time2str(Timestr,'24','hm','hm')])
  ylabel('Lift/Cargo Asset Status')
  xlabel('Lift/Cargo Assets')
  drawnow
  subplot(2,1,2)
  TonDisp=[AtBase./MEB;Cargo./MEB;sum(Prep)./MEB;...
    (Warehouse+ESGWarehouse-JOASustain-SupportElement-sum(Prep))./MEB;...
    OnLighter./MEB;Delivered./MEB;...
    [JOASustain(1:2)./(sum(DailySustain(:,1:2))*DaysSustain),0]];
  TonDisp=round(TonDisp.*100);
  bar(TonDisp,'group')
  axis([.5 6.5 0 100])
  title(['Lift Efficiency: ' num2str(round((TotalLift-TotalLiftWait)/...
    TotalLift*100),'%.0f') '%    Lift Error: ' ...
    num2str(LiftErr,'%.0f') 'min'])
  ylabel('% of MEB Deployment')
  xlabel('Location of Cargo')
  set(gca,'XTickLabel',{'AtBase','Cargo Ship','In Prep','On ITS',...
    'On Lighter','On Shore','Sustain'})
  set(gca,'XTick',[1 2 3 4 5 6])
  legend(char('Tons','sq. ft.','People'),-1)
  pause(1)
end
```

```
end
%
% Objective Metrics
Output.DeliveryTime=Time/TimeSteps;
Output.LiftEff=(TotalLift-TotalLiftWait)/TotalLift*100;
Output.CargoEff=(TotalCargo-TotalCargoWait)/TotalCargo*100;
Output.LiftErr=LiftErr;
Output.MaxWarehouse=MaxWarehouse;
for i=1:size(Lighter,2)
  Output.LiftSortie(i)=sum(([LiftAsset.Type]==i).*([LiftAsset.Sortie]));
end
j=1;
for i=1:1:length(CargoShips)
  CargoUsage(i)=sum([CargoAsset(j:j+CargoShips(i).Number-1).Sortie]>0);
  j=j+CargoShips(i).Number;
end
Output.CargoUsage=CargoUsage;
```

Page Intentionally Left Blank

Appendix I: ITS Optimization and Model Description

While the ITS is primarily a single mission platform, it has a variety of characteristics that set it apart from other platforms. It must have the airlift capability of an LHA, the cargo carrying characteristics of RO/RO and container ships. It has well deck interfaces similar to LPD and LSD class amphibious ships and must support a large number of personnel. Different mixes of personnel, cargo carrying capabilities, transfer interfaces, and assembly areas all can affect the mission performance.

Many would point to the fact that large ships typically have increasing returns to scale in terms of cargo carrying capacity of the ship since capacity is a function of volume ($length^3$) while total operating cost may be less than a function of volume due to less hull surface, nonscaled officers and crew, etc. This would lead to the use of fewer numbers of larger ships to carry cargo, the typical trend in the shipping industry. On the other hand, cargo throughput is also dependent on the interface points which is either a function of surface (e.g. the flight deck is a function of length*beam) or even a function of length (e.g. side interfaces are a function of length, well interfaces a function of the beam) (Figure I1) thus the interface points could result in decreasing returns of scale. These opposing trends to size may result in nonintuitive results regarding the size and scale of each platform depending on the magnitude of the increasing and decreasing returns to scale. This means that the ITS must be examined within the context of the larger system which uses aspects of decreasing returns to scale to identify an optimal design.

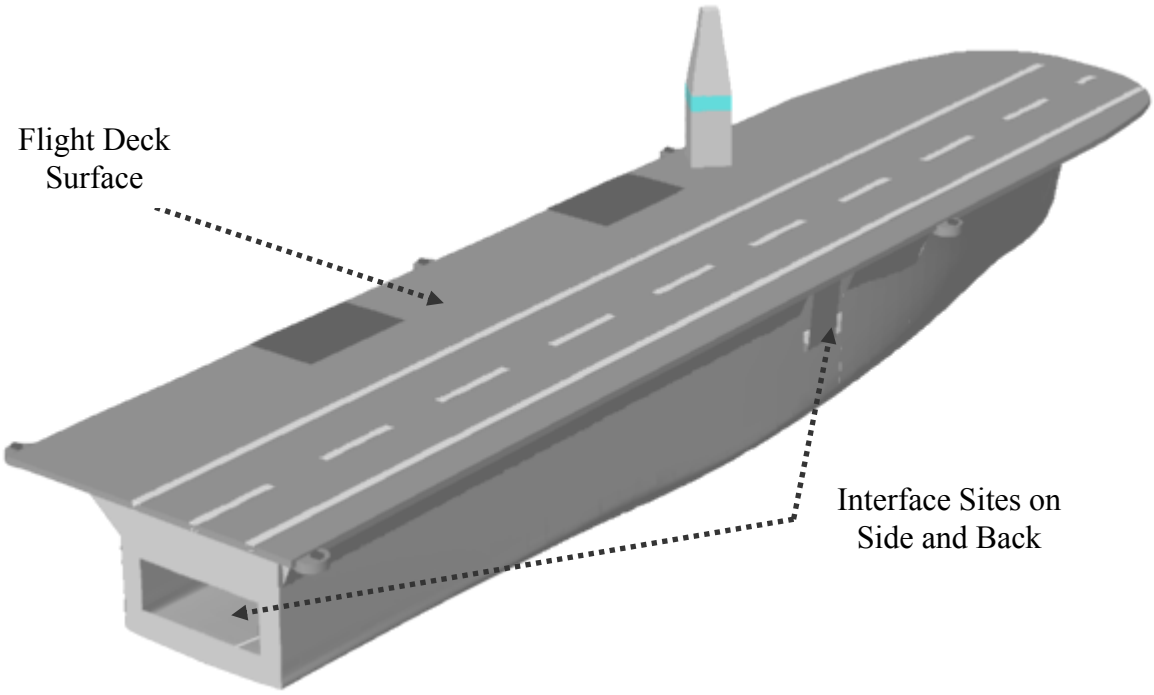


Figure I1. Conceptual Drawing of ITS Showing That Interface Sites Are Functions of Length and Surface

ITS Optimization

The optimization of the ITS design is done using the MATLAB function **fmincon** with a medium scale search. The medium scale **fmincon** algorithm uses a SQP optimization method. SQP is a gradient based numerical optimization method for constrained continuous nonlinear

problems. The basic SQP method involves taking an initial solution and developing a quadratic approximation of the objective function. To account for the nonlinear constraints, the Lagrangian is used in lieu of the objective function. In accordance with the Kuhn-Tucker Conditions, Equation (I1), one necessary condition for optimality is that the gradient of the Lagrangian must equal zero. A quadratic subproblem is developed by approximating the Lagrangian, using a second-order Taylor series expansion, and constraints, using a first order Taylor series. The quadratic subproblem is solved using a quadratic programming method to determine the search vector which optimizes the quadratic subproblem. A lines search along the direction of the search vector is performed to arrive at a new solution. The Hessian of the Lagrangian is updated and the process is repeated. The mathematics of the SQP method as well as descriptions of how to solve a quadratic program, perform a line search, and update the Hessian matrix are described in more detail in (Gill et al., 1981).

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \cdot \nabla G_i(x^*) = 0$$

$$\nabla G_i(x^*) = 0 \qquad i = 1, \ldots, m_e$$

$$\lambda_i^* \geq 0 \qquad i = m_e + 1, \ldots, m \qquad \text{(I1)}$$

where x* is the optimal solution
f(x) is the objective function
λ_i are the Lagrange multipliers corresponding to constraint i
G_i are the i constraints of the problem

The formulation of the ITS optimization problem is shown in Equation I2. Details of the objective and constraints follow.

min	J = Displacement = f(**X**)	
s.t.	**g**(**X**,**Y**,**Z**) ≤ 0	
D.V.	**X** = [L, B, D]'	(I2)

where X is the vector of subsystem design variables
Y is the vector of system level compatibility variables
Z is the vector of system level performance variables
g(**X**,**Y**,**Z**) is the set of constraints
L is the ship length
B is the ship beam
D is the ship depth

ITS Ship Objective

The objective of the ITS Model and Optimization is to minimize displacement. Equation (I3) is used to calculate the displacement as a function of length, beam, and depth.

$$\text{Displacement} = L * B * D * C_b * T/D * \nu \qquad \text{(I3)}$$

where L is the ships length
B is the ship beam
D is the ship depth
C_b is the block coefficient (Assume .6)
T/D is the draft-to-depth ratio (Assume .6)
ν is the specific volume of seawater (35 ft^3/LT)

ITS Ship Constraints

Conceptual ship design typically begins with identifying the key limiting characteristics; namely, determining if the ship is weight or volume limited. Weight limited design implies the model should carefully balance the available weight as it is more restrictive then volume. Conversely, volume limited design carefully balances the available volume rather than weight. Most ships including warships are volume limited, with notable exception being large bulk and crude oil. To determine whether the ITS is weight or volume limited, Watson describes a process to make the determination by essentially comparing the cargo density (cargo weight/cargo volume) to the ship density (~displacement/hull volume) (1998). If the cargo density is greater than the ship density, then the ship is weight limited. Otherwise, it is volume limited.

For the sea base, it is necessary to have access to the various cargo elements for loading or unloading as needed. Having access for this selective loading requires less dense stowage which creates significant open spaces within the ship. Personnel and other office spaces which will occupy a significant portion of the ship also reduce the effective cargo density. These two effects along with the fact that passenger and container ships are almost always volume limited lead to the assumption that the ITS is volume limited and thus weight balance is not required.

Available Versus Required Space

While the ITS is volume limited, the cargo model used in the performance simulation is based on rolling square feet. Therefore a relationship must be developed between ship volume and available square footage. Since the ITS has different capabilities than any other existing platform, it is difficult to derive an empirical relationship using other ships. Fortunately, the MPF(F) performs many of the same missions. Data from ships considered in the MPF(F) Analysis of Alternatives (AoA) were examined. To calculate the square footage available for troops, interfaces, and cargo, data for each of these elements was converted to square feet using Equation I4 (i.e. each interface required X square feet for an assembly area, personnel each require Y square feet for berthing, messing, and habitability, etc.)

$$\text{Cargo Area} = 37.5*X_1 + 7000*X_2 + 4500*X_3 + 3.3*X_4 + X_5 \qquad \text{(I4)}$$

where X_1 is the number of troops onboard
X_2 is the number of interface spots
X_3 is the number of LCACs carried
X_4 is the weight of fuel
[3.3 ~ 43 ft^3/LT / 14 ft deck height *1.07 for tank structure expansion]
X_5 is the amount of cargo square

The total volume was calculated from the length (L), beam (B) and draft (T) using Equations (I5), (I6), and (I7) from (Watson, 1998).

$$C_{bT} = \Delta / (L * B * T) \quad \text{(I5)}$$

$$C_{bD} = CbT + (1 - CbT) * (0.8 * D - T) / 3 / T \quad \text{(I6)}$$

$$\text{Volume} = L * B * D * CbD \quad \text{(I7)}$$

where C_{bT} is the draft block coefficient
Δ is the ship displacement
L is the ship length
B is the ship beam
T is the ship draft
C_{bD} is the depth block coefficient
D is the ship depth (estimated to be T/.33 similar to LHA)

A power regression resulted in a relatively close approximation to the data and is shown in Figure I2.

There are two primary problems with this regression. First, the regression does not cover the entire range of values for possible ships (up to volumes of 18 million ft^3 within shipyard and ratio constraints discussed below). Second, none of the ships used in the regression have actually been built. Therefore, to confirm the results, an analysis was done of RO/RO ships. RO/RO ships were chosen since they are well established as a sea going vessel, there is a large number of them commercially available, and rolling cargo space could theoretically be used for other purposes.

An empirical Equation (I8) obtained from (Lamb, 2003), was used to correlate the volume to lane length. The lane length was multiplied by a typical lane width of 3.5m to obtain an available area.

$$L\,(m) * B\,(m) * D\,(m) = 27 * \text{Lane Length}\,(m) \quad \text{(I8)}$$

The results of using Equation I8 on the MPF(F) ships is also shown in Figure I2.

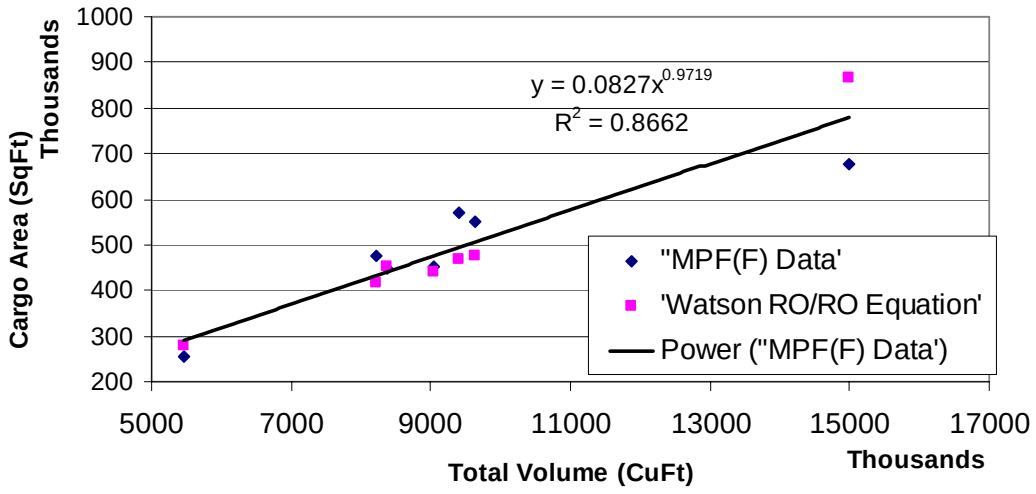


Figure I2. MPF(F) Cargo Area vs. Total Volume

In general Equation (I8) fits the regression analysis very well. Since it is easier to optimize using Length, Beam and Depth, Equation (I8) was used to determine available arrangeable cargo space for the ITS.

The available space calculated using Equation (I8) is assumed to be usable for any purpose. For instance, it could be a berthing area, an assembly space, or a well deck. While this may be appropriate for a concept design, much greater concern for space arrangement must be applied during subsequent design. However, this method allows each of the required spaces to be accounted for together and compared against the available space.

Accommodations for the crew would not be part of the lane length and so are assumed to be accounted for in the remainder of the ship. Accommodations for troops carried aboard the ITS are of two classes, one for troops that stay on board for an extended period such as combat support personnel, referred to as the support element, and one for troops that surge through the ITS and remain onboard for less than one week. The support element is assumed to have accommodations which meet U.S. Navy standards for embarked troops. Common spaces shared with crew are assumed to meet Military Sealift Command habitability standards. Therefore, embarked troops are given an average of 130 ft^2 each. The surge troops are assumed to have austere accommodations. They are given the same sanitary facilities, but are expected to be berthed in tighter quarters. Food service and leisure areas are also not provided for these personnel. Therefore, each surge troop has an average area of 50 ft^2. Note that these values are averages and different personnel such as officer and enlisted would require different area allocations. For this analysis, it was assumed one officer for every nine enlisted. The required personnel space was further modified by a factor (9/14) to account for different deck heights used in personnel spaces and RO/RO cargo lanes.

Each interface spot is allocated 7,000 ft^2 for an assembly area. This value is similar to that used in the MPF(F) designs. Additionally, each LCAC well deck spot is assumed to take 9,000 ft^2 of space (90ft x 50ft x 2 decks). An additional 20,000 ft^2 is required for lighter maintenance if organic well deck craft are present. If organic aircraft are present, hangar space of 50,000 ft^2 is required along with ~13,000 ft^2 per aircraft. However, the ITS is not a command and control ship, therefore spaces related to this function were not included.

The final space consideration concerns ballast and fuel tankage. A certain amount of ballast is already accounted for in the ship to maintain stability. However, additional ballast to lower the well deck to the waterline is not. Most likely, the ship would be trimmed by the stern to lower the well deck opening. However, due to the difficulty in calculating the Moment to Trim 1 Inch (MTI) because of the uncertainty with centers of gravity, buoyancy and metacentric height, the ITS is assumed to ballast down two feet using parallel sinkage. Therefore the required additional ballast volume is calculated using Equation (I9), which is converted to ft^2 by assuming a 14ft deck height

$$\text{Ballast} = (L*B*Cw/420) * X * \nu * k \qquad \text{(I9)}$$

where $(L*B*C_w/420)$ is the Tons Per Inch immersion (TPI)
L is the ship length
B is the ship beam
C_w is the waterplane coefficient (Assume .75)
X is the additional parallel sinkage in inches (Assume 24 in)
ν is the specific volume of seawater (35 ft^3/LT)
k is a tankage adjustment factor for structure and expansion (~1.07)

Likewise, cargo fuel tankage is determined based on the MEB requirements and aviation and well deck refueling to be conducted at each interface spot. The following assumptions correlate the interface spots and number of days of sustainment to the cargo fuel required on board.

- 400 LT per day for MEB ground personnel
- 28 LT/day/Well Deck Interface for LCAC & HLSL operations
- 10.3 LT/day/Air Interface for MV-22/CH-53 operations

After all these spaces have been accounted for, the remaining space is assumed to be available for cargo stowage. A stowage factor is applied to this amount to account for how densely packed the cargo might be. Previous studies have shown a stowage factor of approximately 50% is suitable to selective off loading of cargo (Souders et al., 2004). The remaining space with the stowage factor must exceed the goal cargo capacity to be a feasible design

Site and Lighterage Constraints

The number of each interface site type is also checked to see if the ship is large enough to accommodate. The side sites are properly handled by the system level boundaries (i.e. up to two, one for each side) The air interface sites must fit on the unobstructed flight deck with each site being large enough to accommodate a CH-53 (the worst case clearance requirement). The number of potential spots is calculated using Equation (I10) and must exceed the goal air sites to be a feasible design. A value of 1.1 is used for k to account for added sponsons which increase the effective flight deck beam and for clearance areas which may overhang the deck edge as well as reduced area taken by the flight control tower. (Based on data for LHD-8, the k value would be 1.05 without accounting for clearance overhang)

$$AirSites = L * B * k / (\pi * (d/2)2) \qquad (I10)$$

where L is the ship length
B is the ship beam
k is unobstructed flight deck area/(L*B)
d is the aircraft clearance diameter (~115 ft for CH-53) (Naval Air Systems Command, 2003)

Based on the ITS length and beam, the number of equivalent LCAC well deck sites is calculated. Both the ITS beam and length are adjusted by factors to determine the maximum size well deck. The maximum potential LCAC equivalent well deck sites must exceed the goal well deck sites to be a feasible design.

$$WellSites = k_1 * (B/50) * k2 * (L/90) \qquad (I11)$$

where k_1 is the Well Deck Beam-to-Ship Beam Ratio (Assume .8)
(B/50) is the number of LCAC the ITS is wide
k_2 is the Well Deck Length-to-Ship Length Ratio (Assume .6)
(L/90) is the number LCAC the ITS is long

In addition to the number of sites, the potential well deck length (.6*L) and beam (.8*B) are checked against the HLSL size to ensure that it would fit within the available well deck. Finally, the total potential well deck area (.8 * B * .6 * L) is checked to make sure that all organic LCAC and HLSL would fit. For simplicity, Equation (I12) was used to calculate the required well deck area. It does not take into account the actual arrangement of the lighters or the possibility of unusable space within the well deck. The potential well deck area must exceed the requirement to be a feasible design.

$$\text{WellArea} = \sum L_i * B_i * n_i \qquad \text{(I12)}$$

where i is either the HLSL or LCAC
L is the length of the LCAC or HLSL
B is the beam of the LCAC or HLSL
n is the number of organic LCAC or HLSL

Stability and Size Constraints

To ensure the ITS is stable, within typical ship dimensions, and has the ability to be built within existing shipyards, several additional parameters were checked. For stability, a common conceptual design practice is to use the metacentric height-to-beam ratio (GM/B). An arbitrary but historically adequate range is .09 to .122. Because of the low fidelity of this model, the allowable range was extended to .085 to .1225 to allow for greater optimization flexibility, understanding that more detailed design can alter the value of GM. To calculate GM, equations (I13), (I14), and (I15), were used (Gilmer & Johnson, 1982).

$$KB = (D/5) * (2.5 - C_b / C_w) \qquad \text{(I13)}$$
$$BM = L * B^3 * CIT / (8 * L * B * D * Cb) \qquad \text{(I14)}$$
$$GM = KB + BM - KG \qquad \text{(I15)}$$

where KB is the vertical center of buoyancy
D is the ship depth
L is the ship length
B is the ship beam
C_w is the waterplane coefficient (Assume .57)
C_w is the waterplane coefficient (Assume .75)
BM is the metacentric radius
C_{IT} is the transverse inertia coefficient (-.497 + 1.44 * C_w)
KG is the vertical center of gravity (Assume D/2)

Typical ship dimensions were also checked by ensuring various dimension ratios were within values typical of ocean going cargo ships. Table I1 gives the ranges used for these ratios. The ranges used were obtained from Watson (1998).

Table I1. Feasible Range of Dimension Ratios

Ratio	Lower Bound	Upper Bound
L/B	5	8
B/D	1.4	1.95
L/D	10	12

Finally, the actual dimensions were checked against shipyard sizes to ensure at least one would be large enough to build the ITS without having to build a new dock. The maximum combinations of length and beam are shown in Table I2.

Table I2. Representative Shipyard Dimension Limits

Shipyard	Length (ft)	Beam (ft)
A	1020	174
B	1084	146
C	994	171

Validation

To check the validity of the model the results of an optimization run were compared to those obtained during a design project to create an ITS using more sophisticated design tools (Johnson et al., 2005). The inputs to the model are given in Table I3 and a comparison of results is given in Table I4. At first glance, with errors as high at 16% it would seem that the model does not perform very well. However, when looking at the ratios of length-to-depth (L/D) and beam-to-depth (B/D), we see that both of these are below the minimum levels of 10 and 1.4 respectively which would make the design project ITS an infeasible design in this ITS Model. In order for the ITS Model to optimize to a feasible design, depth had to decrease and beam increase, which they do to keep L/D and B/D near their minimum values. Violating the L/D and B/D constraints make stability more difficult to achieve during detailed design due to the higher center of gravity which is related to depth. It is also important to note that the design project ITS hull form was not optimized and thus should be expected to have a larger L*B*D value which it does.

Table I3. Validation Inputs

Variable	Value
Capacity (Sqft, Pers)	[125000, 3500]
Interface Sites (air, well, side)	[12, 2, 1]
Lighter (LCAC, MV-22, CH-53, HLSL)	[8,0,0,0]

Table I4. Validation Result Comparison

Dimension	13A Design Project ITS	Model Results	% Error
Length (ft)	1000	1002.1	.2%
Beam (ft)	138	147.5	6.9%
Depth (ft)	120	100.2	-16.5%
L*B*D (ft^3)	1.66e7	1.48e7	-10.8%
L/D	8.3	10	
B/D	1.15	1.47	

Appendix J: ITS Optimization and Model Code

```
function [SysErr, Ceq, Disp]=ITSCOOptimizer(Input, Param)
%
% The purpose of this script is to optimize the ITS within constraints
%
% Objectives: 1 - meet system level inputs
%             2 - minimize displacement (a proxy for cost)
% System Variables: Number, Capacity, Sites, Lighter, LighterSize (B&L)
% Subsystem Variables: Length, Beam, Depth
% Constrants: within existing shipyards, stability
%
% Objective Function: J=C*Length*Beam*Depth
%
% Inputs:   Input.Name     = Name (not adjusted)
%           Input.Speed    = Speed (not adjusted)
%           Input.Range    = Range (not adjusted)
%           Input.Number   = Number
%           Input.Capacity = Capacity [LT, sqft, pers]
%           Input.Sites    = Transfer Sites [Air, Side, Well]
%           Input.Lighter  = Organic Lighters [LCAC,LCU,CH-53,MV-22,SeaBAC]
%           Input.AAAV     = Organic AAAV (not adjusted)
%           Input.LighterB
%           Input.LighterL
%
% Outputs:  SysErr      = sum of constraint violations squared
%           Ceq         = []
%           Disp        = J
%
% Created by Robert Wolf, - Jan 2005
%

load ITSOptimData % Get data from previous optimizations
Data=[Input.Number, Input.Capacity(2) Input.Capacity(3), ...
    Input.Sites(1:3), Input.Lighter(1), Input.Lighter(3:5),...
    Input.LighterB, Input.LighterL]; % Setup comparison data variable
SimFlag=1; % Set flag
for x=1:1:size(ITSOptimData,1) % Check previous optimization data
    if all(ITSOptimData(x,1:12)==Data) % Is this optimization not unique?
        X=ITSOptimData(x,13:15); % Get optimization solution
        Disp=ITSOptimData(x,16); % Get optimization result
        SimFlag=0; % Skip Optimization
        break % Stop looking for same optimization in file data
    end
end

if SimFlag % Unique optimization
  x0=[1082, 146, 80]; % Initial Starting Point (L,B,D)
  lb=[100,10,10]; % Lower bounds on L,B,D
  ub=[1082,174,100]; % Upper bounds on L,B,D
  % Options for fmincon
  Options.ActiveConstrTol= [];
  Options.DerivativeCheck= 'off';
  Options.Diagnostics= 'off';
  Options.DiffMaxChange= 0.1000;
  Options.DiffMinChange= 1.0000e-008;
```

```
  Options.Display= 'off';
  Options.GradConstr= 'off';
  Options.GradObj= 'off';
  Options.Hessian= 'off';
  Options.HessPattern= 'sparse(ones(numberofvariables))';
  Options.LargeScale= 'off';
  Options.MaxFunEvals= 10000;
  Options.MaxIter= 4000;
  Options.MaxPCGIter= 'max(1,floor(numberofvariables/2))';
  Options.MaxSQPIter= Inf;
  Options.PrecondBandWidth= 0;
  Options.TolCon= 1.0000e-006;
  Options.TolFun= 1.0000e-006;
  Options.TolPCG= 0.1000;
  Options.TolX= 1.0000e-006;
  Options.TypicalX= 'ones(numberofvariables,1)';
  % Optimization
  [X,Disp,exit,output,lambda]=fmincon(@ITSObjective,x0,[],[],[],[],...
    lb,ub,@ITSCOFun,Options,Input);
  ITSOptimData=[ITSOptimData;Data, X, Disp]; % Add this data to file
  save ITSOptimData ITSOptimData; % Save optimization data
end

[C,Ceq]=ITSCOFun(X,Input); % Get constraint information
SysErr=sum(max(C,zeros(size(C))).^2); % Calculate sum of errors squared
if SysErr<1e-10;SysErr=0;end % Adjust to account for optimizer tolerance




function [J]=ITSObjective(x,Param)
J=x(1)*x(2)*x(3)*.6*.6/35/10000;
```

```
function [C,Ceq]=ITSCOFun(Input, Param)
%
% The purpose of this script is to analyze the ITS for its constraints
%  Optimality is determined seperately by minimizing displacement
%
% Input:    Input(1)        = Length
%           Input(2)        = Beam
%           Input(3)        = Draft
%
% Param:    Param.Name      = Name (not adjusted)
%           Param.Speed     = Speed (not adjusted)
%           Param.Range     = Range (not adjusted)
%           Param.Number    = Number
%           Param.Capacity  = Capacity [LT, sqft, pers]
%           Param.Sites     = Transfer Sites [Air, Well, Side]
%           Param.Lighter   = Organic Lighters [CH53,MV22,LCAC,LCU,SeaBAC]
%           Param.AAAV      = Organic AAAV (not adjusted)
%           Param.LighterL
%           Param.LighterB
%
% Outputs:  C(1)            = SqFt Capacity constraint
%           C(2)            = Wt Capacity constraint
%           C(3:5)          = Sites Constraints
%           C(6:7)          = Lighter Size Constraints
%           C(8)            = Organic Lighter Space Constraints
%           C(9)            = Stability Constraint
%           C(10:11)        = L/B Constraints
%           C(12:13)        = B/D Constraints
%           C(14:15)        = L/D Constraints
%           Ceq             = []
%
% Sizing and weights are approximate and derived from various unclassified
% US Navy documents
%
% Created by Robert Wolf – Jan 2005
%
SqFtPerUSMC=50;
SqFtPerSupport=130;
SqFtPerSite=7000;
SqFtPerWell=90*50*2; % Size of LCAC (nominal 2 decks)
SqFtPerAir=0; %Air is topside thus no internal volume unless hangar
SqFtPerSide=0;  % on side, so no internal volume
SqFtTotCommand=0; % Command space on different ship
AirFuel=775;     % Air fuel per Day (tons)
% Assume fuel per site based on CH-53 equivalents
% Find CH-53 equivalents by taking CH-46 equivalent size of MEB (199)
% Divide by CH-53 size in terms of CH-46s (2.68)
% To get the CH-53 equivalents (~75)
% Thus each Air spot should take about AirFuel per day/ 75
AirSiteFuel=AirFuel/75;
MEBFuel=400;     % MEB Fuel per Day (LT)
LCACFuel=28;     % Fuel per LCAC per day (LT)
TankageAdjust=1.05*1.02;
TonsPers=.35;
DaysSustain=5;
CbGuess=.6;
TGuess=31;
```

```
UsableDeck=1.1; % % of L*B usable for flight deck (Assuming Sponsons)
UsableSide=.7;
UsableWell=.5;
WellLen=90;
SideLen=190;
SuppPers=760;
VolLimit = 4890000; % Based on Kavaerner Shipyard Size
%
Cw=.75;
L=Input(1);
B=Input(2);
D=Input(3);
TPI=L*B*Cw/420;
ShipyardConstraint=[ 1020 174 50000;
                     1082 146 72400;
                      994 171 30000];
%
% ***** SqFt Capacity Error Check *****
% Based on RO/RO from Watson (/33 factor accounts for m to ft conversion)
LaneLength=L*B*D/27/33;
SqFtAvail=LaneLength*3.5*10; % Based on Lane Width of 3.5m or ~35 ft
if sum([Param.Lighter(3:4)])~=0
    SqFtAir=50000+115^2*sum([Param.Lighter(3:4)]); % Aircraft Hangar
    % 50000 is maintenance space from MPF AoA, 115^2 is for aircraft
    % storage, 500 is for additional personnel berthing and office space
else
    SqFtAir=0;
end
if sum([Param.Lighter(1:2)])+Param.Lighter(5)~=0
    SqFtSurf=20000; % Surface craft maintenance
    % 20000 is a pure guess on maintenance space, 500 is for personnel
    % berthing and office support.  No stowage required within well deck
    % constraint (outlined below)
else
    SqFtSurf=0;
end
% Personnel Areas.  Note 9/14 factor accounts for difference between
% lane length deck(14') and other decks(9') Note: MSC personnel not
% included since assumed part of non-lane length area
StowageFactor=.5;
SqFt=SqFtAvail-...     % Available Area
    SqFtPerSite*(sum([Param.Sites]))-... % Interface Assembly Areas
    SqFtTotCommand/Param.Number-...     % Command and Control / Maintenance
    SqFtPerWell*Param.Sites(3)-...     % Well Deck
    SqFtAir-SqFtSurf-...                % Organic Lighter Support
    (SqFtPerSupport*SuppPers+...
       SqFtPerUSMC*(Param.Capacity(3)-SqFtPerSupport))*9/14-...
    % LT cargo fuel per day converted to sqft (assume 14' deck height)
    (AirSiteFuel*Param.Sites(1)+LCACFuel*Param.Sites(3)+MEBFuel)*...
       (DaysSustain*43/14*TankageAdjust)-...
    (TPI*24*35/14)*TankageAdjust; % Ballast Tanks (2' parallel sinkage)
C(1)=(Param.Capacity(2)-SqFt*StowageFactor)/Param.Capacity(2);
%
% ***** Weight Capacity Error Check *****
% Ship is probably volume limited, so LT capacity error based on if cargo
% weight greater than percentage of max displacement
HullVol=Param.Capacity(2)/(.01669+3.6748e-07*Param.Capacity(2));
```

```
VolDisp=HullVol/35;
PersWt=TonsPers*Param.Capacity(3);
FuelWt=(AirSiteFuel*Param.Sites(1)+LCACFuel*...
  (Param.Sites(2))+MEBFuel/Param.Number)*DaysSustain*TankageAdjust;
CargoTot=Param.Capacity(1)+PersWt+FuelWt;
% Regression based on similar functioned ships
WtDisp=CargoTot/(.0706+1.2285e-05*CargoTot);
LightShip=max(WtDisp,VolDisp)-CargoTot;
SizeError=-any(all(repmat([L,B,LightShip],3,1)<ShipyardConstraint));
C(2)=SizeError;
%
% ***** Air Site Error Check *****
% Calculate the possible air sites based on CH-53 size
%  Assume clearance circle of 115ft (from NPS report on seabase ship)
%  Assume flat top deck
%  Note: Clearance over deck edge, LOA>LBP, and tower assumed to even out
AirSites=L*B*UsableDeck/(pi*(115/2)^2); % round down to integer
C(3)=Param.Sites(1)-AirSites;
%
% ***** Side Site Error Check *****
% Side sites are to be limited to 2
C(4)=Param.Sites(3)-2;
%
% ***** Well Site Error Check *****
% Calcultate the possible well deck sites based on LCAC
%  Assume stern gate can cover 80% of beam
%  Assume well deck can cover 60% of the length
%  LCAC Well Deck Spot of 50ft x 90ft
%  Well Deck assumed longitudinal with FILO capability
WellSites=.8*B/50*min(floor(.6*L/90),1);
C(5)=Param.Sites(2)-WellSites;
%
% ***** Lighter Size Error Check *****
%  Length and Beam should accomodate within well deck
C(6)=(Param.LighterL-.6*L)/L;
C(7)=(Param.LighterB-.8*B)/B;
%
% ***** Well Deck Lighter Capacity Error Check *****
%  Should have no more Surf Lighters than Sites
%    Verified by simple area calc
% Fit SeaBAC (HLSL) first
WellSize=.8*B*.6*L;
SeaBACArea=Param.LighterB*Param.LighterL;
WellSizeRem=WellSize-SeaBACArea*Param.Lighter(5);
LCACArea=50*90;
%
C(8)=max(Param.Lighter(2),0)+...
    max(Param.Lighter(5)-WellSize/SeaBACArea,0)+...
    max(Param.Lighter(1)-WellSizeRem/LCACArea,0);
%
%
% % ***** Stability Error Check *****
T=D*.6;
KG=D/2;
CIT = -0.497 + 1.44*Cw;
KB = (T/3)*(2.5 - ((.6 * .95)/Cw));
BM = (L*B^3) * CIT/(12 * L*B*T*CbGuess);
```

```
GM = KB + BM - KG;
CGMB = GM/B;
C(9)=max(.085-CGMB,CGMB-.1225)/.1;
%
% ***** L/B Ratio Check *****
% L/B of 5 to 8 Watson p. 67
% B/D of 1.4 to 1.95 with most ships at top end
% L/D of 10 to 12 (MIT math model limits to 15 for structural reasons)
C(10)=5-L/B;
C(11)=L/B-8;
C(12)=1.4-B/D;
C(13)=B/D-1.95;
C(14)=10-L/D;
C(15)=L/D-12;
Ceq=[];
```

Appendix K: HSC Model and Optimization Description

The HSC platform was optimized using a hull form comparison tool developed by the Maritime Applied Physics Corporation (MAPC). The limited program documentation states that the results are not meant for design purposes and represent only one of many possibilities. Thus the results from the MAPC model do not represent an optimized design in the truest sense, but rather a balanced design which meets the various goals. However, the model was used because it was already available, illustrates how to interface with another program, approaches design in a different way than the other models (via goal programming), matches how the standard collaborative optimization framework works, and allowed multiple different hull forms to be examined. No validation of the MAPC model result was performed for this research as it was assumed to have been done when the model was written.

The HSC model starts the MAPC model and sets goals for Speed, Payload, and Range. The values for speed and payload came from the system level variables. Range was fixed to be 2000nm, a nominal distance between the AB and the sea base. Another required input is to prioritize the three goals in the order the MAPC model should achieve them. Once the #1 priority goal is met, the MAPC model begins trying to increase goal #2. Since Range was a fixed quantity, it was given first priority. From there, two runs were done prioritizing range, speed, payload and range, payload, speed in those orders.

With the goals and priorities, the MAPC model begins with several basis vessels of the same hull form. The formulation for each hull form is very similar with the difference related to the results of empirical relationships for the weight fractions of weights other than fuel and payload as well as values of block coefficient, length-to-beam ratio and beam-to-draft ratio. Empirical relationships are also made for specific fuel consumption as a function of horsepower and required shaft horsepower/displacement as a function of Froude number (a nondimensional quantity of speed, gravity, and length). An additional factor is also calculated to take into account the reduction in speed due to rough weather which is applied to the calculated speed.

The balancing begins by adjusting the fuel to meet the range. As fuel is increased, the displacement goes up requiring more horsepower to reach speed. As horsepower goes up, the specific fuel consumption goes down requiring less fuel. Additionally, the higher horsepower may require a different engine. Increasing speed increases the Froude number which increases the required horsepower for the displacement. More horsepower lowers specific fuel consumption but more fuel is still required because of the increased horsepower to meet the range. Increasing payload increases displacement which increases required shaft horsepower which lowers the specific fuel consumption but still increases required fuel because of the higher horsepower required. The ship is further constrained by a maximum displacement of 4000LT and the use of 2 LM6000s which provide up to 114660 shaft horsepower.

Each of these three effects is solved iteratively by adjusting fuel for range, speed and payload in priority order. Once the iterations fall below the desired accuracy, the model stops. Length, beam and depth are calculated using the final displacement and the empirical values for block coefficient, length-to-beam ratio, and beam-to-draft ratio.

The HSC model retrieves the final attained values for speed, payload, and range as well as the final displacement, length and beam. The length and beam are used to check that the potential cargo area, Equation K1 exceeds the system level variable for square foot capacity.

$$\text{HSC Cargo Area} = k * C_b * L * B \qquad \text{(K1)}$$

where k is the stowage factor (assume .5)
C_b is the block coefficient (assume .6)
L is the length of the HSC
B is the beam of the HSC

Three different hull forms are evaluated, mono hull, catamaran, and trimaran. For each of the six designs (two different priority orders for each of the three hull forms), the negative errors between the system level variables and the HSC design results (i.e. where the HSC design did not meet the performance goal) are summed. The six designs are then sorted based on this error and the displacement. The design with the smallest displacement and zero error is selected. If no design had zero error, then the design with the smallest error is selected.

Appendix L: HSC Optimization Code

```
function [C,Ceq,Output,Vehicle]=HSC_CO_Optimizer(Input, Param)
% This script is used as the HSC module for collaborative optimization
% *** This function must be run with the mapc directory in the MATLAB
% working directory for the function to work
%
% Objective Function: min   J=Displacement
%                     s.t.  Goal Speed <= Achieved Speed
%                           Goal Payload <= Achieved Payload
%                           Goal sqft <= Achieved sqft
%                     D.V.  # and type of Engines, Displacement,
%                           Hullform type (Monohull, catamaran, trimaran)
%
% This optimization works by interfacing with the MAPC Hullform analysis
%   tool, written in EXCEL to determine an appropriate size and ensure the
%   feasibility constraints are met. MAPC is a goal programming optimizer
%   thus goals are given and priorities assigned to each goal.  For this
%   project, each set of priorities is run to find the best overall design.
%
% The best design first minimizes errors in achieving goals.  If all goals
%   are met on more than one design, then the design with the smallest
%   displacment is selected
%
% Inputs:   (Part of the inputs to performance module
%           Input.Speed     = Speed
%           Input.Capacity  = [Payload, SqFt, Pers]
%
% The Outputs are C and Ceq (constraint values) and Output
%   C -  sum of overage errors in Speed and Capacity
%   Ceq = []
%   Outputs - Displacement
%
% Note: Payload in MAPC is assumed equal to cargo tonnage
%       L*B*.3 is used to calculate available sq.ft.
%           .3 chosen because Cp of .6 for area at waterline
%           Assume 50% of area is usable

% Setup variables
directory=[cd,'\mapc\'];
filename=char(['catamaran.xls'],['MONOHULL.xls'],['Trimaran.xls']);
SortedRows=[];
Goals=[Input.Speed, Input.Capacity(1), Input.Capacity(2), Input.Range] ;

% Check to see if the data has been optimized before
load HSCOptimizedData; % Location of previous optimization results
for i=1:size(HSCOptimizedData,1)
    if all(HSCOptimizedData(i,1:4)==Goals) % Same as previous data
        SortedRows=HSCOptimizedData(i,5:11); % Get results
        break % stop looking
    end
end
%
if isempty(SortedRows) % No previous optimization on this data?
    % Open an Excel Server.
    Excel = actxserver('Excel.Application');
```

```
    Workbooks = Excel.Workbooks;
    %
    Atotal=[3,2,1;2,3,1]; % Two goal programs(Speed-Payload, Payload-Speed)
    % The goals: Range is fixed input, 4000 is max displacement for MAPC
    B=[Input.Speed;Input.Capacity(1);Input.Range;4000];
    count=0;
    for y=1:1:2 % try both goal programs
        A=Atotal(y,:)';
        for x=1:size(filename,1)
            count=count+1;
            Workbook = invoke(Workbooks, 'Open', ...
              [directory,filename(x,:)], 1); % 1 forces update
            Sheets = Excel.ActiveWorkBook.Sheets;
            InterfaceSheet=get(Sheets,'Item',6); % 6th sheet in workbook
            CalcSheet=get(Sheets,'Item',5); % 5th sheet in workbook
% set(Excel, 'Visible', 1); % uncomment if you want to see the worksheet
% invoke(InterfaceSheet, 'Activate');% Make the interface sheet active.
% Activesheet = Excel.Activesheet;% Get a handle to the active sheet.
            %
            % Get various Excel arrays
            PriorityRange = get(InterfaceSheet,'Range','B4','B6');
            GoalRange = get(InterfaceSheet,'Range','D4','D7');
            SpeedRange = get(InterfaceSheet, 'Range', 'D10');
            PayloadRange = get(InterfaceSheet, 'Range', 'D11');
            RangeRange = get(InterfaceSheet, 'Range', 'D12');
            ResetRange = get(InterfaceSheet, 'Range', 'D17');
            DispRange = get(InterfaceSheet, 'Range', 'D13');
            CostRange= get(CalcSheet, 'Range','C79');
            LengthRange= get(CalcSheet, 'Range','C60');
            BeamRange= get(CalcSheet, 'Range', 'C61');
            %
            % Set goal and priorities
            set(ResetRange, 'Value', 'Y');
            set(PriorityRange, 'Value', A);
            set(GoalRange, 'Value', B);
            set(ResetRange, 'Value', 'N');
            % Get back a range.  It will be a cell array, since the range
            % can contain different types of data.
            Results(count,1)=SpeedRange.Value;      % Speed
            Results(count,2)=PayloadRange.Value;    % Payload
            Results(count,3)=.3.*LengthRange.Value.*BeamRange.Value; % SqFt
            Results(count,4)=RangeRange.Value;      % Range
            Results(count,5)=DispRange.Value;       % Displacement
            % Error which only looks at how far below goal the ship is
            Error=(Results(count,1:4)-Goals)./Goals; % Normalize errors
            Results(count,6)=dot(Error,sign(Error)-1)/2; %sum the errors
            Results(count,7)=x; % Hullform type
            % To avoid saving the workbook and being prompt to do so,
            Workbook.Saved = 1;
            invoke(Workbook, 'Close');
            invoke(Excel, 'Quit');
        end
    end
%
% Pick a best design (sort by error then displacement)
[SortedRows, indexRows]=sortrows(Results,[6,5]);
% Add this data to the file
```

```
HSCOptimizedData(size(HSCOptimizedData,1)+1,:)=[Goals, SortedRows(1,:)];
delete(Excel); % Release the ActiveX server interface
end
%
% Errors for Optimization
C=SortedRows(1,6); % Sum of overage errors
Ceq=[];
Output=SortedRows(1,5); % Best displacement
% Actual data (not used)
Vehicle.Speed=SortedRows(1,1);
Vehicle.Capacity=[SortedRows(1,2),SortedRows(1,3),Input.Capacity(3)];
Vehicle.Range=SortedRows(1,4);
Vehicle.Disp=SortedRows(1,5);
Vehicle.Type=filename(SortedRows(1,7),1:8);

save HSCOptimizedData HSCOptimizedData; %Save previous optimization data
```

Page Intentionally Left Blank

Appendix M: HLSL Model Description

The HLSL model and optimizer were used to design and select an ACV. Design tools for this type of craft are extremely limited and proprietary thus instead of designing an ACV, the model selects from a set of existing or potential designs. The design points were obtained from two sets of data. The first was from an MIT new ship construction design project which ran a design of experiment run of data using software designed by a private firm on contract to the U.S. Navy. Because these designs were from a model used for ACV design within the U.S. Navy, no additional validation was done and the designs were assumed to be feasible solutions. The second set of data was from existing ACVs in operation today (Miller & Gougliodis, 2005). These two sets of data were combined and the non dominated designs were identified based on maximizing speed and payload, and minimizing length, beam, and displacement.

The HLSL model and optimizer selects the design which at least achieves the goal for speed and payload and whose length and beam are greater than the actual length and beam but within a 20% tolerance. Since length and beam are coupling constraints, it was important to ensure that they were not unbounded. However, the conceptual level of the design would make designing to an exact common length and beam impractical, thus the 20% margin. Of the designs which meet the above constraints, the one with the smallest displacement is selected. This process can be illustrated graphically in Figure M1.

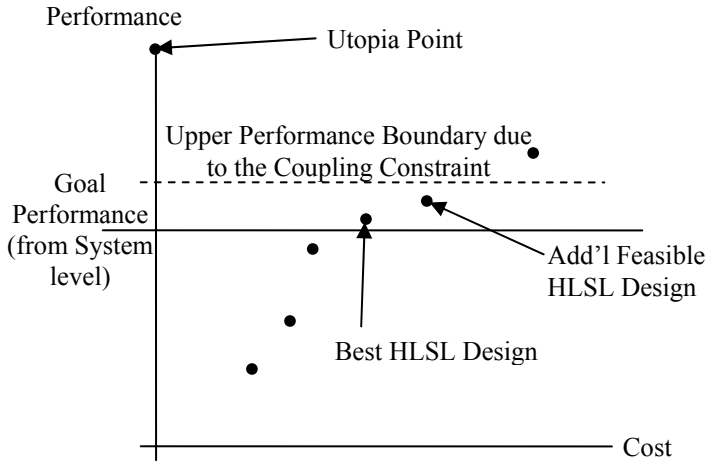


Figure M1. HLSL Design Selection

The HLSL model and optimizer does not actually use any subsystem design variables, however, there are several beyond the performance goals that are implicitly included in the designs, such as number and type of engines, skirt pressure, etc. It is assumed that these designs were each optimized to minimize displacement (cost) while meeting various constraints (stability, strength, weight balance, etc.).

Page Intentionally Left Blank

Appendix N: HLSL Optimization Code

```
function [C,Ceq,Outputs]=SeaBAC_CO_Optimizer(Input, Param)
%
% The purpose of this function is to select the best air cushioned craft
%
% The optimization is done based on two sets of data.  The first is from
% the SeaBAC Design team from the MIT 13A program (Brian Miller, George
% Gougoulidis).  This data was based on design runs using software designed
% by a private firm on contract with the Navy.  The data is assumed to be
% correct and provide feasible solutions.  The data was sorted to give
% non-dominated solutions in terms of speed, payload, and size. The second
% set of data is from the SeaBAC report on existing air cushioned vehicles.
%
% The two sets of data were combined to create a Pareto-optimal set from
% which to select.
%
% Objective Function: min Displacement
%                     s.t. Goal Speed <= Actual Speed
%                          Goal Payload <= Actual Payload
%                          Goal Length & Beam > Actual Length & Beam
%                          Goal Length & Beam < Actual Length % Beam *1.2
%                     DV: Not explicitly examined in this algorithm but
%                        included # and type of engines, as well as
%                        constraints, stability, etc.
%
% Inputs:   (Part of the inputs to performance module (+ Length & Beam)
%           Input.USpeed    = Speed
%           Input.Capacity  = [Payload, SqFt, Pers]
%           Input.Length    = Length
%           Input.Beam      = Beam
%
% The Outputs are C and Ceq (constraint values) and Output
%   C -  sum of overage errors in Speed, Capacity, Length, and Beam
%   Ceq = []
%   Outputs - ACV Disp
%
% Data:         Spd Pyd L   B   H       Disp
% based on Pareto sort of SeaBAC Data
Data=       [
   43.0000   25.0000   25.2000   11.7000   55.0000
   55.0000   50.0000   32.0000   15.0000  150.0000
   45.0000   54.3000   26.8000   14.3000  153.5000
   70.0000   90.0000   47.8000   17.5000  260.0000
   45.0000  135.0000  140.0000   65.0000  510.5980
   60.0000  135.0000  130.0000   70.0000  517.7270
   60.0000  130.0000   57.6000   25.6000  550.0000
   40.0000  203.0000  170.0000   65.0000  673.4430
   50.0000  203.0000  150.0000   70.0000  675.0390
   60.0000  203.0000  150.0000   75.0000  677.1910];
% Add the 20% extra length and beam
 Data=[Data(:,1:2) -Data(:,3:4) Data(:,3:4).*1.2 Data(:,5)];
%
% Find the min for the ACVData
Goals=[Input.USpeed, Input.Capacity(1), -Input.Length, -Input.Beam,...
  Input.Length, Input.Beam];
```

```
%
x=(repmat(Goals,size(Data,1),1)-Data(:,1:6));
y=x.*(sign(x)+1)./2;     % eliminate satisfied constraints
[z,index]=min(sum(y'));  % smallest disp satisfying constraints
%
C=z; % minimum error
Ceq=sum([Data(index,1:6)-Goals].^2); % Total Error
Outputs=Data(index,7); % Best ACV Disp
```