Fuzzy Optimal Allocation and Arrangement of Spaces in Naval Surface Ship Design

by

Eleanor Kate Nick

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Naval Architecture and Marine Engineering)
in The University of Michigan
2008

Doctoral Committee:

Professor Michael G. Parsons, Chair
Professor Armin W. Troesch
Associate Professor Kazuhiro Saitou
Professor David J. Andrews, University College London
Professor Bruce C. Nehrling, U.S. Naval Academy

To my soon-to-be husband, Dave.

ACKNOWLEDGEMENTS

Thank you to my research team: Mike Parsons, Jignesh Patel, Dave Singer, Hyun Chung, Su Liu, and Tony Daniels. Tony, we have certainly learned a lot together in our years on the Arrangements Optimization team. Parsons, your vision, leadership, and support have been invaluable. I am honored to be your last.

Bob Ames, thank you for being a mentor through my time at Carderock and at Michigan. Jeff Hough, your outreach and energy are unparalleled. Kelly Cooper, you have my great appreciation for sponsoring this research. Thank you also to the folks at ASEE for their support through the DOD (NDSEG) fellowship.

To those who helped me along the way, I give my sincerest gratitude: Leigh McCue Weil, Chris Kent, Steve Zalek, and Weiwei Yu. Leigh, you continue to be an impressive role model. Elisha M.H. Garcia, we are kindred academicians for passing our (second) qualifying exams together! Thank you to my officemates and treasured friends who shared the ups and downs and helped make graduate school a tremendous four years: Leigh, Steve, Jamie Szwalek, Elisha, and Piotr Bandyk.

Thank you many times over to my family. Mom & Dad, you have always helped me to "Be the Best of Whatever You Are". Ben, you lead the way and set the bar. Grammie & Grampa, your unwavering love is truly a source of strength for me. Lastly, thanks to David Kirtley for playing, traveling, understanding, and graduating.

TABLE OF CONTENTS

DEDICATION . . . . . **ii**

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF FIGURES . . . . . **vii**

LIST OF TABLES . . . . . **ix**

LIST OF APPENDICES . . . . . **xi**

CHAPTER

1. Introduction 1

1.1 Overview of Work Done 1
1.1.1 Two Part Approach: Allocation and Arrangement 1
1.1.2 Motivation for Work 2
1.2 Fuzzy Optimization Method: Genetic Algorithm 3
1.2.1 Fuzzy Logic 3
1.2.2 Cost Function Criteria 4
1.2.3 Genetic Algorithm Background 4
1.3 Implementation within Team Project 7
1.4 Organization of Dissertation 11

2. Previous Approaches 12

2.1 Earliest Works 12
2.2 Fuzzy Logic 14
2.3 Grid Fillers 17
2.4 Geometry and Topology 23
2.5 Fuzzy Multi-Attributive Group Decision-Making 26
2.6 SURFCON 27
2.7 Ongoing Work 29
2.8 Shared Approaches 31

3. Part 1: Allocation 34

3.1 Problem Statement 34
3.2 Mathematical Model 37
3.2.1 Independent Design Variables 37
3.2.2 Definition of the Goals and Constraints 37
3.2.2.1 Zone-deck Utility 37

3.2.2.2 Space Global and Relative Location Preferences 38
3.2.3 Definition of the Objective Function 40
3.3 Optimization Method 41
3.3.1 Elitism and Fitness Percentage 42
3.3.2 Tournament Selection 42
3.3.3 Variation Operations 43
3.3.3.1 Simulated Binary Crossover 44
3.3.3.2 Mutation 45
3.3.3.3 Single-Point Crossover 46
3.3.3.4 Pair Swapping 46
3.3.4 Re-seeding and Stopping Conditions 47
3.3.5 Observations from Initial Experiments 48
3.3.6 GenALLOC 48
3.4 Optimization Results 49
3.4.1 Parametric Studies 53
3.4.2 Discussion 54

4. Part II: Arrangement 59

4.1 Two Step Approach: Topology and Geometry 59
4.2 Damage Control Deck Zone-decks 60
4.2.1 Topology 61
4.2.2 Geometry 62
4.2.3 Port and Starboard Sub-Zone-decks 64
4.2.4 Center Sub-Zone-deck 65
4.3 Stochastic Growth Loop 67
4.3.1 Probabilistic Selections 68
4.3.2 Move Rules 71
4.3.3 Appendage Repair Functions 74
4.3.4 Looping Rules and Stopping Conditions 75
4.4 Below Damage Control Deck Zone-decks 75
4.4.1 Stairtowers and Passages From Above 75
4.4.2 Topology 76
4.4.3 Passage Growth Control 81
4.5 Mathematical Model 82
4.5.1 Definition of the Goals and Constraints 82
4.5.1.1 Required Area 83
4.5.1.2 Aspect Ratio 84
4.5.1.3 Minimum Overall Dimension 85
4.5.1.4 Minimum Segment Dimension 86
4.5.1.5 Perimeter 86
4.5.1.6 Adjacency and Separation 87
4.5.1.7 Access Connectivity Separation 88
4.5.2 Definition of the Objective Function 89
4.6 Optimization Method 90
4.6.1 Variation Operations 90
4.6.2 Adaptive Probability Roulette Selection 92
4.7 Example Problem 93
4.7.1 Inputs 93
4.7.2 Results 96
4.7.2.1 Damage Control Deck 96
4.7.2.2 Below Damage Control Deck 99
4.8 Validation Problem 105
4.8.1 Three-spaces 106

4.8.2 Four-spaces . . . 107
4.8.3 Five-spaces . . . 108
4.8.4 Geometry Validated . . . 109

5. Conclusions . . . 111

5.1 Summary . . . 111
5.2 Intellectual Contributions . . . 113
5.3 Future Work . . . 115

APPENDICES . . . **117**

BIBLIOGRAPHY . . . **124**

LIST OF FIGURES

Figure

1.1 Overall Arrangements Optimization schematic 2

1.2 Sample Genetic Algorithm Variation Operations: Mutation and Crossover 6

1.3 Generic Genetic Algorithm 7

1.4 Overall Intelligent Ship Arrangements schematic 9

2.1 An example of Lee et al.'s (2002) facilities layout and corresponding representation of the four-segmented chromosome 20

2.2 A feasible concept design by Van Oers et al. (2007) 31

3.1 Example Ship Inboard Profile 35

3.2 Gaussian-Based Model for Zone-deck Area Utilization Utility U_{Zd_k} 38

3.3 Allocation Genetic Algorithm Schematic 41

3.4 Sample Genetic Algorithm Variation Operation: Two-space Swap 47

3.5 Allocation Inboard Profile 49

3.6 Histogram and Analysis of Individual Space Utilities U_{spaces} 52

3.7 Convergence Plots of Allocation Parametric Studies 54

3.8 Effectiveness of GenALLOC on Early Population Fitness 55

4.1 Arrangements Optimization schematic 60

4.2 Example ISA Damage Control Deck Passages and Stairtowers 60

4.3 DCD Topology Centroids Mapped to the Zone-deck 62

4.4 Space Representation using Three Boxes 64

4.5 Center Box Initial Expansion in the Center Sub-Zone-deck 66

4.6 Stochastic Growth Loop Schematic 67

4.7 Center Box Attachments Example Steps 1 and 2 79
4.8 Center Box Attachments Example Steps 3 and 4 79
4.9 Center Box Attachments Example Steps 5 and 6 80
4.10 Example Piecewise Linear Fuzzy Utility 83
4.11 Default Required Area Fuzzy Utility 84
4.12 Sample Shape with Overall Dimensions and Minimum Segment Dimension MSD . 84
4.13 Default Minimum Overall Dimension Fuzzy Utility 85
4.14 Default Minimum Segment Width Fuzzy Utility 86
4.15 Sample Proximity Distance by Closest Points 88
4.16 Separation Distance Between Two Accesses on DCD 89
4.17 Arrangements Optimization Expanded Schematic 91
4.18 Convergence Plot for DCD GA 97
4.19 DCD First (a and b) and Final (c and d) Solutions showing Adjacencies (white) and Separations (black) 98
4.20 Cost Function and Elapsed Time versus Maximum Geometry Generation Iteration for Case 1 and 2 100
4.21 BDCD Parametric Study: CF vs Remaining Spots 101
4.22 BDCD Parametric Study: Remaining Spots vs Iteration 102
4.23 Below DCD Solution 1, CF = 0.8956 103
4.24 Below DCD Solution 2, CF = 0.8855 103
4.25 Below DCD Repaired Solution 2, CF = 0.9179 104
4.26 Three-space $U = 1$ Validation Solutions 106
4.27 Four-space Validation $U = 1$ Solution (a) and Alternate Solution, $U = 0.9892$ (b) . 108
4.28 Five-space $U = 1$ Validation Solution 109
A.1 Abridged Best Chromosome History with Creating Operations, Part 1 of 3 119
A.2 Abridged Best Chromosome History with Creating Operations, Part 2 of 3 120
A.3 Abridged Best Chromosome History with Creating Operations, Part 3 of 3 121

LIST OF TABLES

Table

2.1 Author and Methods Comparison 33

3.1 Global Location Preference Fuzzy Utilities for Space Groups 39

3.2 Relative Location Preference Fuzzy Utilities for Space Groups 40

3.3 Allocation Space Preference Summary 50

3.4 Zone-deck Area Allocation Results 50

4.1 Sample Space J Geometry Variable Matrix 63

4.2 Probabilities for Space Growth Values -3 to +3 69

4.3 Percentage Probabilities for Grow and Shrink Growth Directions by Aspect Ratio . 71

4.4 Control Points for a Piecewise Linear Fuzzy Utility 83

4.5 Required Area Fuzzy Utility Default Control Points 84

4.6 Aspect Ratio Fuzzy Utility Default Control Points 85

4.7 Minimum Overall Dimension Fuzzy Utility Default Control Points 85

4.8 Minimum Segment Dimension Fuzzy Utility Default Control Points 86

4.9 Perimeter Fuzzy Utility Default Control Points 87

4.10 Adjacency and Separation Fuzzy Utility Function Default Control Points 89

4.11 Access Separation Fuzzy Utility Default Control Points 89

4.12 Number of Swaps and Crossovers per Generation by Zone-deck type 92

4.13 Allocation Solution applied to Arrangement example problem 94

4.14 Connectivity Matrix for DCD 95

4.15 Area Satisfaction from First and Final DCD Arrangement 97

4.16 Area Satisfaction from Below DCD Arrangements 104
4.17 Three-space Validation Problem Inputs 106
4.18 Four-space Validation Problem Inputs 107
4.19 Five-space Validation Problem Inputs 108
4.20 Validation Problem Statistics 109
B.1 Final Allocation Configuration: Individual Space Utilities, Part 1 of 2 122
B.2 Final Allocation Configuration: Individual Space Utilities, Part 2 of 2 123

LIST OF APPENDICES

Appendix

A. Abridged Best Chromosome History with Creating Operations 118

B. Final Allocation Configuration: Individual Space Utilities 122

CHAPTER 1

Introduction

1.1 Overview of Work Done

Presented in this dissertation is a new approach to generating, evaluating, and optimizing general arrangements of naval surface ships. Beginning from a user editable database of spaces, the proposed algorithm returns an optimized arrangement. The user drives the design by quantitatively defining spaces' goals and constraints for location, adjacency, and shape. A fully autonomous problem solver is neither feasible nor desirable. The goal is to provide the naval architect with a semi-automated tool.

1.1.1 Two Part Approach: Allocation and Arrangement

The arrangements task is undertaken in two parts: Allocation and Arrangement. Allocation is the assignment of a space to a region of a ship. The unit region used is dubbed a Zone-deck. The Zone-deck is the intersection of one deck and one subdivision. This is essentially a very large scale combinatorial bin packing problem. With the added complexity of relative location constraints, this also becomes a type of quadratic assignment problem.

Next, the second part handles one Zone-deck at a time defining its assigned spaces' joiner bulkhead locations. Arrangement is done is two iterative steps: topology and geometry. Topology gives the relative fore and aft position of each spaces' seed location. These locations are translated onto an orthogonal grid. In the second step,

they are expanded to have size and shape filling the available area of the Zone-deck in a stochastic growth loop. The best of a modest number of geometry solutions returns all the bulkhead locations on the grid and a cost function evaluation of the solution. This double loop structure is shown in Figure 1.1.

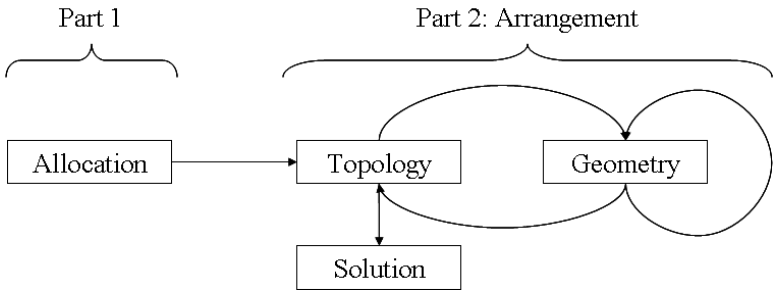


Figure 1.1: Overall Arrangements Optimization schematic

1.1.2 Motivation for Work

There is currently no precise assessment of arrangements. They are typically generated by a naval architect with years of experience dictating what works. Rules and criteria for these ad-hoc methods are documented, but they do not include all of the consideration needed for a strong arrangements solution. General arrangements remains very much an art. As the workforce and its accumulated experience ages, it is important to capture these methodologies for preservation. This requires the difficult task of translating the art and science of arrangement into a series of rational decisions. Further, through implementation in a computer program, the optimization can be linked to the extensive database of goals and constraints imposed by requirements and best practice. With thousands of constraints needing to be simultaneously considered, the need for computational optimization is clear. A further goal of this project is to write a cost function to validate solutions beyond their present subjectivity. The advantage of making smart decisions earlier in design is frequently addressed. Changes made to the design later become increasingly more

costly. This tool will allow the designer to generate and evaluate more solutions that are closer to optimal quickly and easily in the preliminary design stage.

1.2 Fuzzy Optimization Method: Genetic Algorithm

1.2.1 Fuzzy Logic

Ship arrangement is characterized by a large number of vaguely defined, conflicting and subjective considerations, opinions, and preferences as well as a plethora of explicit requirements and constraints. As first advocated by Nehrling (1985), fuzzy logic is an ideal way to evaluate all the cost function goals and constraints. Criteria measures are translated to fuzzy utility values via fuzzy membership functions. The domain of the functions are the possible parameter values for the preference in question; they can be either discrete, such as for the preference for a space to be located in one certain Zone-deck on the ship, or continuous, such as for a constraint for a radar room to be within the maximum wave-guide length of its associated antennae. The range of the fuzzy membership functions is a continuous scale between zero and one. Zero corresponds to unsatisfactory. One is perfectly satisfied. A utility of 0.8 might signify "great", while 0.6 is "good". With the freedom given to the user of editing fuzzy preference values and membership functions, the user has the capability to give inputs that truly reflect the design intent.

There are multiple benefits to applying fuzzy logic. The trade-offs naturally occurring in any interesting design problem can be modeled. A less than perfect utility for one preference might have to suffice if another preference is to remain viable. Fuzzy logic captures this imprecise nature while still allowing clear comparison on the normalized range of zero to one for the many goals and constraints of various units and scales. The effect of each variable can be uniquely studied. Lastly, fuzzy logic pairs well with optimization. It is undesirable to return an infeasible solution

result and stop. By keeping low utility values, optimization can improve the solution, rather than exit early with no results. A Boolean expression of only zero or one for arrangements would be inadequate as there is likely no perfect ship arrangement.

1.2.2 Cost Function Criteria

Many cost function criteria are applied to test for satisfaction. In the allocation problem, Zone-deck area utilization and space location preferences are examined. All available area should be assigned to spaces. Spaces also have preferences to be in certain regions of the ship and preferences for adjacency and separation to other spaces. In the arrangement problem, space shape factors are also considered. Aspect ratio, minimum overall dimension, minimum segment dimension, and perimeter length are considered along with the required area. Connectivity to access is established and maintained. Again in Part II, adjacency and separation between spaces is considered, but at the finer scale within the Zone-decks.

1.2.3 Genetic Algorithm Background

Solutions are optimized using Genetic Algorithms, GAs. GAs are modeled after biological processes found in genetics as indicated by the vocabulary. They are particularly useful for searching very large and sparse domains by operating on a population of solutions (Goldberg 1989; Gen and Cheng 1997). Allocation and topology variables are combinatorial and discrete, and there is no gradient in the search space to direct solutions toward an optimum as used in many other optimization algorithms. The stochastic nature of a GA combines elements of existing solutions to produce improved solutions. GAs are effective in finding solutions scattered in this type of multi-modal domain.

In the overall work flow of the arrangements optimization task (Figure 1.1), a GA is first used to find an optimal solution for the allocation problem. Second, a

separate GA is applied to the current Zone-deck's topologies. Because a topology can produce many geometries, n number of geometries are generated for each topology in a stochastic process. The best geometry is determined by enumeration. After the optimization of topologies, the best geometry for the best topology is the solution.

A Genetic Algorithm operates on a collection (a population) of solutions at once. Each solution is a variable vector called a chromosome. A variable expressed in the chromosome is called a gene. Genes may take on different values; these are called alleles. For example from biology, on a chromosome there might be a gene for eye color. Two possible alleles are blue and brown. For allocation, the gene value expresses in which Zone-deck the associated space is located, and the alleles are the Zone-deck indices. The allocation chromosome lists all the spaces' Zone-deck locations in order by space index.

As a human's DNA changes, so too do these chromosomes. To hopefully find better chromosomes, they undergo variation operations. The original chromosome(s) is called the parent. The chromosome(s) created in the operation is the daughter. The two most common operations are mutation and crossover. A mutation randomly alters one randomly selected gene of one chromosome to a different allele. Crossover takes two chromosomes and exchanges gene segments between them at a randomly determined sever point. In simple crossover, the Daughter_1 is the union of the Head of Parent_1 and the Tail of Parent_2. Similarly, Daughter_2 is the Head of Parent_2 and the Tail of Parent_1. Figure 1.2 gives a simplistic view of these operations using binary chromosomes with five genes that may each express maize or blue.

The basic sequence of a possible simple Genetic Algorithm is shown in Figure 1.3. First an initial population of chromosomes is generated and evaluated in the cost function. If elitism is applied, the best (elite) chromosome is tracked for preserva-

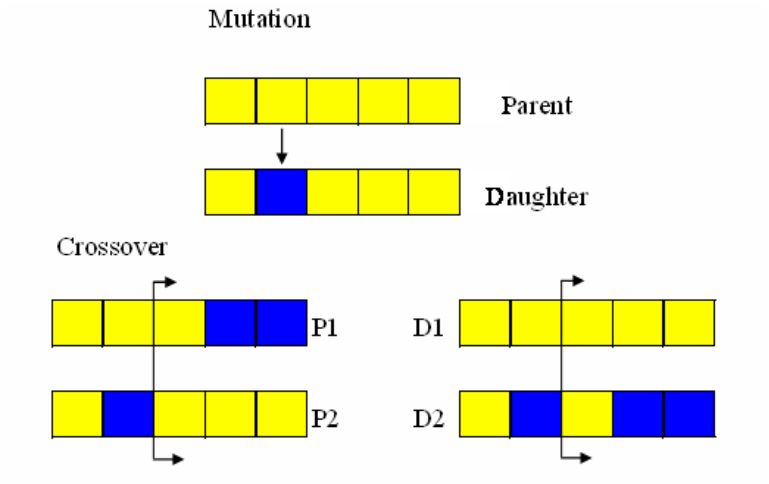


Figure 1.2: Sample Genetic Algorithm Variation Operations: Mutation and Crossover

tion. After each variation operation, the daughter chromosome(s) typically replaces the parent(s) in the population. Using each chromosomes' cost function value, the selection operation probabilistically chooses the more fit chromosomes to survive to the next generation, Figure 1.3.

The most common selection mechanism is Roulette Selection. Each of the possible choices has a certain probability percentage to be chosen ($Psel$). In this case, cost function values are summed and normalized over the total to find a percentage. Each choice is assigned a cumulative probability value equal to the sum of the preceding percentages and its own percentage ($Rsel$). The last choice's cumulative probability value, or its roulette probability value ($Rsel_{last}$), is 1, and $Rsel_1 = Psel_1$ for the first choice. A random real number between zero and one is drawn. If the random number is less than the roulette probability value of Choice$_i$ but greater than the $Rsel_{i-1}$, then Choice$_i$ is selected. Within the allocation and arrangement GAs, selection is carried out as many times as the number of chromosomes in the population to entirely re-write the new generation.

Generations continue until a stopping condition has been reached. Stopping con-

ditions may be a threshold fitness for the elite chromosome, maximum number of generations, or by some other mechanism. At exit, the most fit chromosome is the optimized solution.

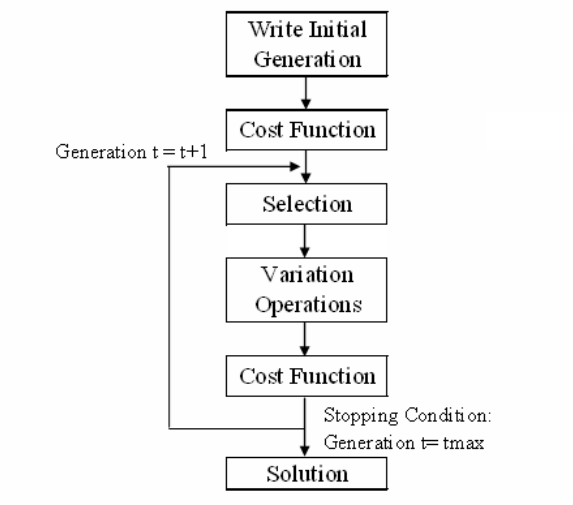


Figure 1.3: Generic Genetic Algorithm

A Genetic Algorithm does not guarantee a global optimum, the absolute best of any feasible design, but it can usually achieve the global optimum with a high probability. When investigating such a large combinatorial search space, it is not reasonable or sometimes even possible to find a perfect solution. Arrangements have a large search space because of the many different ways there are to arrange the spaces that will reside in a full-scale ship. By operating on a population of solutions at once, the GA does a better job of approaching the global optimum in a multimodal search space than an optimizer working with one solution at a time. In this research, the genetic algorithm proved to be effective in generating and improving upon arrangement solutions.

1.3 Implementation within Team Project

This work is part of a team project developing a full-scale prototype software for use within the Navy's design environment, LEAPS (Leading Edge Architecture for

Prototyping Systems) (Ames and Van Eseltine 2001). Intelligent Ship Arrangements (ISA) beta version is to be released shortly after the publication of this dissertation. An overall work flow is presented in Figure 1.4. The main contribution to the software is the work contained here describing the algorithms developed for arrangements optimization.

Central to the methodology of the arrangements optimization project is the development of its space database. Upon program initiation, the user first loads one of the available space list templates keyed to a ship-type. The project currently supports a corvette template based upon a JJMA, now Alion Science, provided Notional Corvette. Templates document all spaces expected on the ship type by Ship Space Classification System (SSCS) number, name, and quantity. Spaces included in the particular ship type template are a subset of the available space library supported by ISA.

Spaces come with default constraints extracted from NAVSEA 070 guidelines (Naval Sea Systems Command 1992), classification society regulations, and from designer experience. Spaces and constraints may be added or edited by the user through the Constraint Editor window. Constraints are expressed as fuzzy utility functions. On the allocation level there are global and relative constraints; functions are discrete by placement into a Zone-deck. Global position is with respect to absolute location in the ship (i.e. from the bow and baseline). Within ISA, global constraints are handled by assigning utility values to each Zone-deck explicitly. Relative constraints are between two spaces, though the software platform is expandable to relate a space's distance to a fixed location or component, complex of Zone-decks or spaces, or pathway. At the arrangement level, required area, aspect ratio, minimum overall dimension, minimum segment dimension and perimeter are also editable constraints

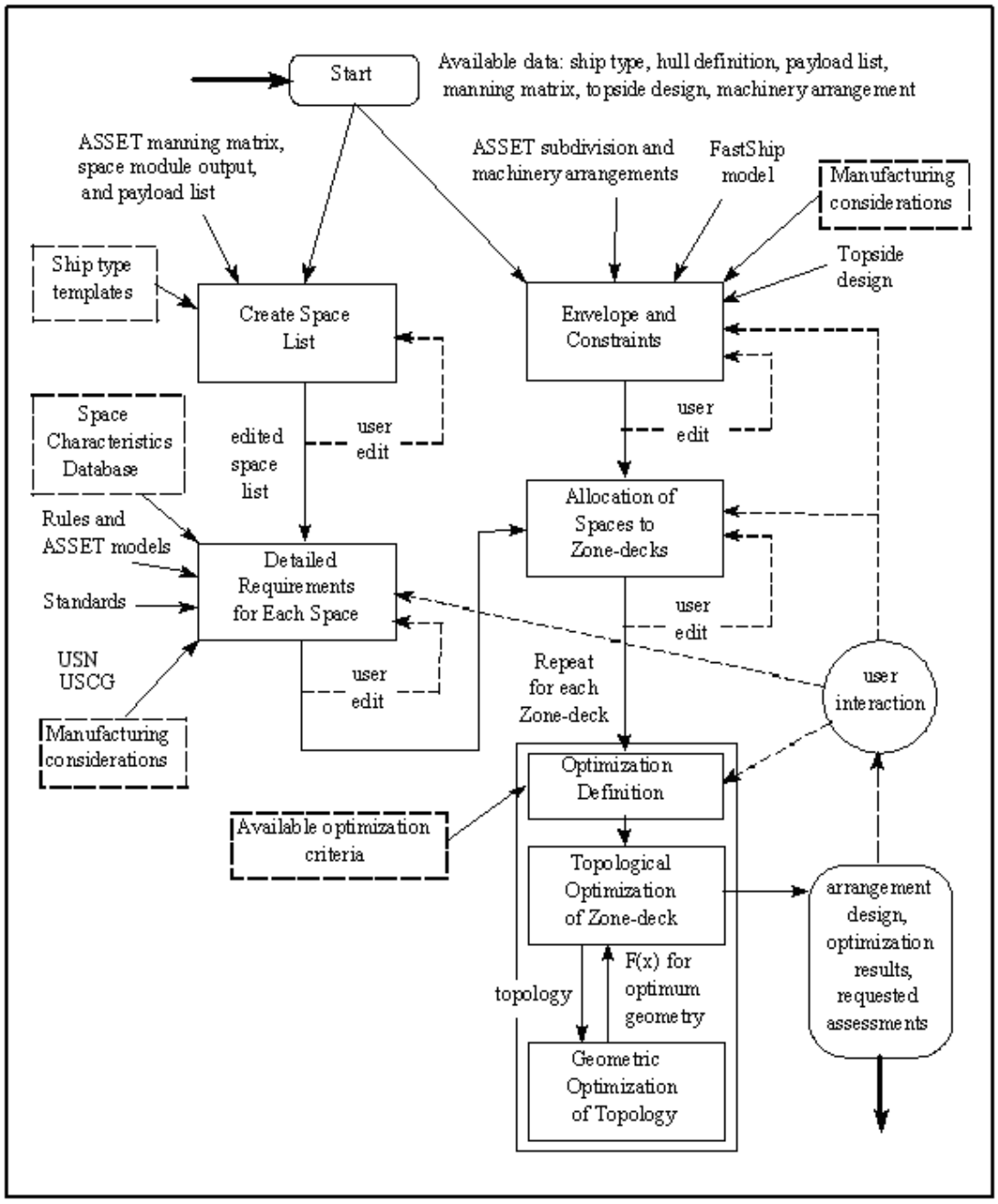


Figure 1.4: Overall Intelligent Ship Arrangements schematic

along with relative position. The space list also states required area and a space index number.

Second, the user imports a LEAPS database. This may contain output from ASSET (Advanced Surface Ship Evaluation Tool) (Naval Surface Warfare Center, Carderock Division 2005) a preliminary design synthesis tool which produces the necessary main structure for ISA: the hull form, decks, and main transverse bulkhead geometry. By a splicing routine, which is part of the LEAPS API (application programming interface), ISA defines the Zone-decks from the known structure. Zone-decks are assigned index numbers used for bookkeeping and optimization. They are not indicative of placement in the ship; the Zone-deck's known subdivision and deck coordinates define its location. A subsequent calculation finds the available area for each Zone-deck. Those that are too small to realistically contain any spaces are marked as exclusion zones. Also, Zone-decks that are fully occupied by machinery or other pre-assigned functions are declared exempt from the allocation.

The Damage Control Deck is designated next by the user selecting all the relevant Zone-decks. Here two longitudinal passages are drawn with continuity along the entire deck. A default two stairtowers are placed within each subdivision. The ISA program allows these to be flipped between the outside and inside of the longitudinal passages. Transverse passages are required at least in alternating subdivisions. It is necessary to recognize the division of the Zone-deck by its passages into Sub-Zone-decks. Spaces can then be allocated to the multiple smaller Sub-Zone-decks rather than the aggregate Zone-deck area. At this point spaces may also be fixed to a Zone-deck. This significant portion of the user interface development has been primarily by Parsons, Chung, Nick, Daniels, Liu, and Patel (2008).

1.4 Organization of Dissertation

This dissertation lays out the presented work in five chapters. In chapter two, previous and ongoing approaches in the field of arrangements optimization give a context for the present work. Chapter three presents the allocation problem. Part II, the arrangements problem, is presented in chapter four. The algorithms, mathematical model, optimization method, and results are described for both parts. A validation problem is also offered at the end of chapter four to illustrate the success of the geometry algorithm. A summary of presented work and future work is included in the concluding chapter.

CHAPTER 2

Previous Approaches

2.1 Earliest Works

The US Navy's attempts to pen a computer-aided ship arrangements program began in the late 1960s. A brief history was given by Carlson and Cebulski of the Naval Ship Engineering Center in 1974 (Carlson and Cebulski 1974). INGAR was the first program. It addressed two-dimensional equipment layout. Next came DEKSUP which allowed a user to drag and drop bulkheads into place for deck and topside design. In 1971, Computer Graphics Arrangement Program (COGAP) was introduced. The scope expanded to hull geometry. Three orthogonal views were available to display the user generated layout. With the subsequent Computer Aided Ship Arrangement Program (CASGAP), Cebulski and Carlson hoped to include optimization and more descriptive graphic input and output in their program. They recognized the need to improve the quantity and quality of design options while still preserving an iterative and user-oriented approach.

CASGAP was composed of ten Modules. Main structural bulkheads were drawn in one. Another module assembled a list of required compartments. This module also takes credit for the first explicit enumeration of compartment requirements. Previously, many requirements were simply assumed or intuited by the designer. These requirements were detailed with size information and adjacency relationships

to other compartments.

The pith of the arrangements design resided in the "Arrange" Module. In two steps, compartments were first assigned to a subdivision and then second defined by bulkhead locations. Compartments were composed of cuboid volumes called "chunks". By chunking, complex compartment shapes were formed with simple volume calculations. In addition to chunk definitions, compartments had these characteristics: Ship Work Breakdown Structure (SWBS) number (Naval Sea Systems Command 1985); required area and volume; required x, y, and z lengths; longitudinal, transverse, and vertical global coordinates; bulkhead requirements; insulation category; access requirement; safety category; group number; and a functional pointer.

Due to the relatively high number of independent variables, optimization was seen as infeasible with the existing processor power. Instead a designer would alter the design slightly and look for improvement. The assessment criteria were not discussed. Hence the iterative design nature was preserved in CASGAP, but neither automation nor optimization was achieved.

The General Arrangements Design System (GADS) was described by Carlson and Fireman, both then working in the Naval Sea Systems Command (Carlson and Fireman 1987). A confluence of conflicting challenges drove this program's creation. Rising costs penalized longer development periods and wasteful designs. Yet finances were increasingly limited. Smart decisions needed to be made quickly, but warships were becoming more complex. A forty to seventy years anticipated life span had to adapt to changing mission requirements and advancing technologies. While automation was needed to reduce iteration time, the designers had to be in the loop as the arrangement process was heavily dependant on human creativity and judgment. However, as a design was continuously updated by different designers, inconsisten-

cies between versions occurred. A single database system was needed to keep all designers involved on the same page.

The GADS system consisted of 29 sub-programs centralized around the Ship Arrangement File (SAF), which acted as a database queried by the Data Access Mechanism (DAM). The SAF was responsible for main ship parameters: hull form, superstructure envelope, bulkheads, decks, access, compartment list, traffic flow, and manning. With the SAF, GADS had the flexibility to break problems into smaller problems using only some of the sub-programs. GADS itself was also a sub-system of the NAVSEA Computer Supported Design system.

Similarly to its predecessor's modules, GADS' sub-programs handled the different steps in an arrangement task, but they reached a much greater level of detail. In addition to compartment requirements, hull form, deck and watertight bulkheads, the topside was arranged, manning and access routes were planned including passages and trunks, and zones (ie. fire and Collective Protection System) were defined. The total list of compartment attributes exceeded forty attributes.

The arrangement was still generated entirely by the user, but the advantage of GADS was that as the arrangement was made, calculations were dynamically performed to alert the user if minimum requirements were not satisfied. For example, bulkhead locations for a compartment could not be placed to enclose less than the required area. Area, access, and V-Line requirements were all automatically checked. Thus, the user could only make viable solutions.

2.2 Fuzzy Logic

To add rational decision making to the arrangement process, Nehrling first advocated the use of Fuzzy Set Theory, a method to evaluate and compare both rigid and flexible goals and constraints (Nehrling 1985). He too saw the drawbacks of the

"non-analytic" and subjective environment in which designers made decisions. With a practically infinite number of possible solutions and a high impact on ship performance, arrangements was a difficult and critical task. Yet, there was no objective definition of a good total arrangement.

As described in chapter one, Fuzzy Set Theory translates all design preferences into fuzzy utility values via fuzzy membership functions. The domain of the functions are the possible parameter values for the preference in question. The range, or possible utility values, of all membership functions is continuous from zero to one. In this manner preferences even with dissimilar units are normalized to the same scale. They can then be compared easily to reveal trade-offs between alternatives.

Nehrling's Fuzzy Membership Functions were written for seven criteria applied to a sample group of twenty ships. All ships were evaluated on: time to general quarters (T [minutes]), habitability index (H [ft^3 per person]), deployment ratio (R [ND]), vulnerability factor (P [ND]), future growth (F [ft]), appearance (A [ND]), and equipment access ratio (E [ND]). The computed and assigned performance values were taken as the arguments for the membership functions. Ship's criteria utility values, U_c, for each criterion c were then combined by taking the intersection, $\bigcap$, representing "AND" to find U_{ship}, eq. 2.1.

$$U_{ship} = U_t \bigcap U_H \bigcap U_R \bigcap U_P \bigcap U_F \bigcap U_A \bigcap U_E = min(U_T, U_H, U_R, U_P, U_F, U_A, U_E) \tag{2.1}$$

The largest U_{ship}, U^*_{ship}, then corresponds to the best design. Nehrling also showed how traditional rigid constraints like the ones implemented in the GADS system were added to the fuzzy membership functions. Any design whose criteria parameter value fell outside the feasible range was then excluded from the set of designs to be considered. Weighting factors on each criteria were also introduced to give greater effect to

those deemed more important. The outcome was sensitive to the new formulation showing the importance of choosing not only appropriate weighting functions, but more so, the shape of the fuzzy membership function.

Shortly after Nehrling's publication, Cort and Hills were the first to utilize his proposed Fuzzy Set Theory approach to rationally evaluate a surface ship arrangement (Cort and Hills 1987). Same as in the GADS system, the hull form was taken as a problem input, and preliminary decks and bulkheads were placed as dictated by other design logic. These structures were again used to delineate zones, but the zone bins were defined by having contents with a common function. Cort and Hills carried out the allocation problem by applying a generic cost function to their multi-objective optimization problem.

Their cost function related distance D with association A added with an environmental loading factor E multiplied by a weighting factor L eq. 2.2.

$$C_{min} = min(\sum\sum A_{ij}D_{ij} + \sum E_i L) \tag{2.2}$$

With this formulation, the desire for adjacency or separation between compartments was captured. The second term penalized against environmental loading such as machinery noise, propeller vibration, or ship motions. Constraints were evaluated with fuzzy logic sets, and then used in the cost function to find an overall cost C for the arrangement.

The optimization was limited to simply choosing the minimum cost function values of various proposed alternative arrangements. However, constraint measures were not normalized, so an additional calculation was made to adjust to a scale from zero (the best) to one (the worst, or maximum cost). Therefore, Cort and Hills' work is not an example of formal optimization, but it is an application of fuzzy set theory to establish an evaluation criterion.

2.3 Grid Fillers

After Fuzzy Logic, the next leap in arrangements' design evolution came from the architecture field by the application of an evolutionary algorithm. Jo and Gero from the Department of Architectural and Design Science of the University of Sydney proposed using a Genetic Algorithm (GA) to handle the combinatorial problem of allocation (Jo and Gero 1998). They noted that for a design problem with n number of spaces, the possible number of ways to put them in a linear sequence was n!. The problem is of the category NP-complete, for nondeterministic polynomial time referring to the required computation time. The computation time does not scale up linearly with additional parameters. In the NP-complete class of problem, it is computationally infeasible to find a global optimum. Instead a set of non-dominated solutions are commonly generated. It is unreasonable to investigate the entire search space, but investigating only one solution at a time is also ineffective. The Genetic Algorithm offers an intelligent way to cover the search space by operating on a collection of solutions at once.

Jo and Gero were optimizing how to best layout spaces in a predefined building envelope. Their spaces had a designated activity and required area. The prototype problem laid out 21 total spaces in a four floor building. Two spaces were fixed and not included in the optimization. The GA chromosome laid out the order in which the remaining 19 spaces were assigned to locations in the building. Spaces' index numbers were written in binary. Five digit genes were used to express numbers zero to eighteen. This ordered list became a popular approach, and would later grow into a formal topology.

An orthogonal grid organized the fixed available internal space. Grid units were called modules, and each activity's required area was translated into a number of

required modules. From the top floor to the bottom, a set path through all modules was defined. According to the order that each activity was listed in the chromosome, the progression through the modules assigned the module's area to the activity until the activity's required number of modules was acquired.

While this formulation did result in feasible spaces, they often had irregular shapes due to the wiggly nature of the assignment path. Also, the one module by one module approach to assigning spaces seems to be unnecessarily slow and arduous when dealing with activity spaces that want larger areas.

After the entire path was traversed through the building assigning area for each activity in the chromosome sequentially, the arrangement was evaluated using a cost function very similar to Cort and Hill's. Global and relative location were not evaluated using Fuzzy Set Theory. Solutions were optimized with a GA.

Single point crossover and mutation operators in the Jo and Gero's GA explored new areas of the search space. However, by the end of 500 generations, there was little diversity left in the population. The best utility was found in the unique chromosomes suggesting that a much greater effort ought to be made in maintaining diversity. Adequate population diversity in the algorithms presented in this dissertation is generated by applying more variation operations and is preserved in the custom selection operations.

The work of Lee, Han, and Roh was motivated by advances in facility layout planning (FLP) of the architecture field, but applied to a naval vessel (Lee et al. 2002). They also used a double summation cost function like Cort and Hill's to capture their multi-objective optimization: minimize transportation cost (f_1) and maximize adjacency (f_2). Weighting coefficients, w_1 and w_2, were multiplied with each objective to give variable representation to the two terms. A third term added internal

penalty functions to the objective function. Since constraints were not expressed in separate functions, the problem is classified as unconstrained. The user wrote upper and lower bound inequality constraints, g_i, for compartment area and aspect ratio. R_u was the corresponding weighting term for each penalty function, u. The unconstrained optimization problem is given in eq. 2.3.

$$\text{Min } F' = w_1 \cdot \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} (f_{i,j} \times d_{i,j}) + w_2 \cdot \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} (C - b_{i,j} \times c_{i,j}) + \sum_{u=1}^{6} (R_u \times \max(g_u, 0)) \tag{2.3}$$

The user predefined: M, the number of compartments; $f_{i,j}$, the material flow between compartments i and j; and $c_{i,j}$, an integer adjacency value between 0 ("undesirable for compartments i and j to be located close together") and 5 ("it is absolutely necessary"). The user also specified the quantity, widths, and lower and upper location limits of the longitudinal and transverse passages. Distances, $d_{i,j}$ were calculated with Dijkstra's algorithm of graph theory. From distance and a maximum distance, the $b_{i,j}$ term was found between 0.0 (far apart) and 1.0 (close together).

The included example problem arranged the area on the Damage Control Deck between three watertight bulkheads enclosing two subdivisions. Two longitudinal passages extended the length of the available area and a transverse passage was placed in each subdivision.

Arrangement of eight compartments was expressed in a four segment chromosome. The first segment gave the order in which compartments would be laid into the available area. Like Jo and Gero, the available area was filled up in a set path. Lee et al. started in the aft-port corner, moved forward, and then continued to the centerline and starboard regions again filling aft to forward across the middle watertight bulkhead. The second segment gave the allotted area for the compartments in the same order as the first segment. Interference between compartments and watertight

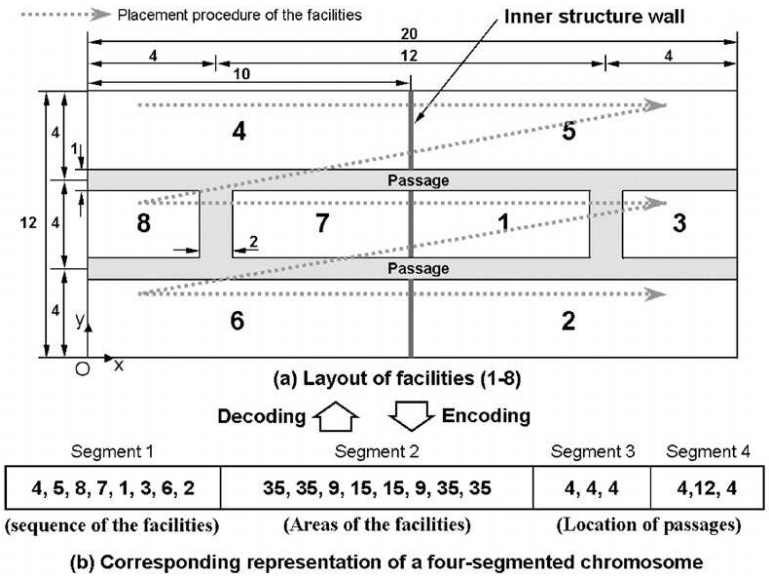


Figure 2.1: An example of Lee et al.'s (2002) facilities layout and corresponding representation of the four-segmented chromosome

bulkheads was avoided by placing the next space in the next region if there wasn't enough room in the current one. The remaining area in the last region was called void space. A refinement operation filled in the void space by assigning it to the compartment already occupying that region. Compartments were all rectangular. No unique geometry or vertical stairtowers were considered. Only area and a rough relative position were included. The third and fourth segments of the chromosome dictated where the longitudinal and transverse passages, respectively, were located. An example chromosome and its corresponding geometry are seen in Figure 2.1.

A Genetic Algorithm was used to optimize the chromosome solution. Crossover and inversion altered the sequence of the compartments and the passages. Mutation acted on the areas in the second segment.

The goal of Lee et al. was to meet the need of better space utilization rising from less dense and larger volume payloads. However, their algorithm would have been unable to deal with an over-allocated area. The areas allotted for each compartment became arbitrary when those values in the second chromosome segment were

allowed to be mutated in the Genetic Algorithm. The compartment's function and its required area are inextricably related. Void space becomes wasted space. It does not appear that they found a way to use the available space optimally. Assuming an equal available hull volume and required spaces' volume defined by the ASSET level output to the arrangements optimization software, the Allocation part solves the space utilization problem in the work presented here.

From the University of Zagreb in Croatia, Slapnicar and Grubisic also used a sequential ordering of compartments in their three step MARRSD (Modular Approach to Ro-Ro Ship Design) approach to layout optimization (Slapnicar and Grubisic 2003). Their problem was divided into three stages: compartment creation, disposition, and valuation. The primary difference of this approach was in stage one where compartment length and width were fixed. Compartments were not allowed to rotate (exchange length and width dimensions). As with previous approaches, these compartments were also rectangular, and their preferred proximity to other compartments, pre-fixed objects, and boundaries were defined in a connectivity matrix during the problem set up.

Concurrent with precedence, Slapnicar and Grubisic laid out their compartments in one zone of the ship at a time with a known shape and overlayed grid. Compartments were placed in the grid by the random order dictated in the compartment list. Their location was determined by a search of available area. Compartments were packed in starting at one corner and filled in bordering the previous so long as there was sufficient room. As with Lee, Han, and Roh, the search for free space proceeded from aft to fore and then transversely. This pattern followed conventional "knapsack" problem methodology. Conversely, the search was on the much smaller grid size rather than by the regions delineated by the longitudinal passages. The

compartment location was defined by the grid points (called knots) occupied.

Layouts were optimized with a closest to utopia calculation. Six attributes were evaluated for each arrangement: area utilization of the zone, distance and communication between compartments, proximity between compartments and pre-fixed objects, proximity between compartments and the outside boundary, and lastly the disposition of all compartments. Recognizing the trade-offs between the objectives, a Pareto-solution was sought. A Pareto-optimal solution cannot improve in one objective without sacrificing performance in another (Papalambros and Wilde 2000). The best (maximum in this case) value achieved for each attribute i is noted as MY_i. The solution offering the best of all attributes was called utopia, UP, eq. 2.4.

$$\mathrm{UP} = (\mathrm{MY}_1,\ \mathrm{MY}_2,\ \mathrm{MY}_3,\ \mathrm{MY}_4,\ \mathrm{MY}_5,\ \mathrm{MY}_6) \tag{2.4}$$

While utopia is not feasible, the goal is to find the solution closest to it. This distance DP in criterion space is calculated for each solution and its attributes' values, PY_i, eq. 2.5.

$$\mathrm{DP} = \sqrt{(\mathrm{MY}_1 - \mathrm{PY}_1)^2 + \ldots + (\mathrm{MY}_6 - \mathrm{PY}_6)^2} \tag{2.5}$$

MARRSD was exemplified for an aft accommodations zone of a Ro-Pax vessel. Twenty-one cabins, four fixed compartments, and seven service compartments were originally placed around two passages. The passages were each represented in two segments for a total of 32 arrangable compartments to be handled by the optimization. A total of 1,434,899 of the possible 32! ($\sim 2.6 * 10^{35}$) variations were generated in 23 minutes. Their final solution, as defined with the initial adjacency preferences, appeared to have strong area utilization. The only apparent drawback was that one of the passages was disjointed. It was treated as two separate components and placed only by searching for available area rather than looking for a logical place to maintain

continuity with the other passage components. While Lee, Han, and Roh gave too much flexibility in their compartments' area, Slapnicar and Grubisic perhaps were too restrictive in their compartments' definition.

2.4 Geometry and Topology

Medjdoub and Yannou approached the architectural layout problem with a novel two step approach: topology and geometry (Medjdoub and Yannou 2000). The topology step emulated the process that an architect goes through to sketch functional relationships between two spaces without consideration of actual dimensions. These topological relationships were shown by graphically connecting nodes representing rooms, cardinal directions, stairs, and corridors.

A full enumeration of all possible combinations of connections was recognized as inefficient. By introducing constraints and negating redundant solutions, the search space was limited to a reasonable size. The remaining topologies were explored by an automated branch and bound optimization within Medjdoub and Yannou's software, ARCHiPLAN. Each node in the tree represented a design choice. If a choice violated constraints, the branch was omitted and a new branch was examined. By a depth-first search all options were attempted. A cost function measured converged solutions. As the search continued, new solutions were pairwise compared to the earlier converged solution to find the better one until the entire tree was explored and a global optimum was revealed. ARCHiPLAN offered more alternatives than the user could generate in comparable time. Should the user want to narrow the choices, additional constraints and criteria could be added, and the topology optimization re-run.

Valid solutions were presented to the user before proceeding to the second step, the geometry definition. From each single topology, Medjdoub and Yannou purported that there were multiple possible geometries. Yet each geometry could be traced

back to uniquely one topology.

Spaces were drawn on an orthogonal grid as rectangles defined by two corner points (X_1,Y_1) and (X_2,Y_2). To form L and T shapes, two rectangles were placed next to each other. The space's length, width, and surface area were given appropriate upper and lower bounds. These constraints defined allowable integer domains for X_1, X_2, Y_1, and Y_2. Aspect ratio limitations further confined these domains. Also, two spaces' rectangles were not allowed to occupy the same area. If the topology made two spaces adjacent, then they had to share a minimum overlapping contact. These constraints were rigid, not fuzzy. Criteria expressed in the cost function were given weighting functions by the user. Medjdoub and Yannou used two criteria, minimal stair and corridor space and minimal total wall length, to show the flexibility of their formulation to handle assorted user preferences.

Optimization was done by enumeration and a branch and bound search. However, increasing the numbers of variables brought an explosion of generated solutions without one designated optimum. Medjdoub and Yannou were only able to tackle what they called a middle-size problem with twenty spaces on two floors. Solutions showed tightly packed rectangular rooms. Only corridors had L and T shapes. Their program showed significant advances in the simplification and organization by approaching the layout optimization problem in two steps.

Michalek, Choudnary, and Papalambros advanced the two step approach in their work at the Optimal Design Laboratory at the University of Michigan (Michalek et al. 2002). Instead of the sequential steps applied by Medjdoub and Yannou, Michalek iterated between topology and geometry. The objective of the topology optimization was to find the best geometry. A Genetic Algorithm was used to optimize the topologies. Only feasible topologies were passed to the geometry step for

evaluation in each generation of the Genetic Algorithm. Geometries were optimized by a hybrid Simulated Annealing and Genetic Algorithm approach. Michalek's architectural optimization problem also expanded the scope of objectives and constraints and re-formulated the independent variables expressed in vector form as $\mathbf{x}$. He was the first to set up a formal optimization problem in standard negative null form (Papalambros and Wilde 2000).

Topologies by Michalek et al. defined connectivities between spaces and spaces' rough location. A connectivity matrix, ϕ_{ij}, expressed the connectivity between space i to space j and to the outer available area boundary in all four cardinal directions where I = total number of spaces for $i = 1, 2, \ldots,$ I and $j = 1, 2, \ldots,$ I, N, E, S, W. If connectivity were required, $\phi_{ij} = 1$, else, $\phi_{ij} = 0$. Topology rough locations were plotted on a grid from two independent integer variable coordinate points (x_i, y_i). Links were added to illustrate the required connectivities between spaces. For a feasible arrangement topology, the links could not intersect; the graph had to be planar.

A topology generated by the Genetic Algorithm that passed the constraints was then evaluated in the geometry step. With this topology definition, each topology could make multiple different geometries, and a single geometry could be made by more than one topology.

Geometry was defined by ten continuous variables for each space. A rectangle was the default shape for each space. More complex shapes were stated to be possible by using two contiguous rectangles, but this case did not appear for any of the living spaces in the presented solutions (Michalek 2001). An arbitrary internal point, (x_i, y_i), was placed on an orthogonal grid. The grid was oriented with the four cardinal directions. Four more variables, N_i, E_i, S_i, W_i, gave the distances to the

corresponding walls from the internal point. Window dimensions were included in the last four variables: $\omega_N, \omega_E, \omega_S$, and ω_W.

There were four types of spaces expressed with these variables: rooms, boundaries, hallways, and accessways. Constraints varied by type. Rooms were not allowed to overlap. Rooms were forced inside boundaries, and accessways were made to intersect rooms. Equality constraints confined rooms to touch boundary edges. Rigid inequality constraints limited minimum area, length, width, and aspect ratio. Lastly, a window had to be smaller than the wall in which it fits. All constraints had to be satisfied to ensure a valid solution.

Solutions were optimized with respect to a multi-objective cost function: min $\mathbf{f}(\mathbf{x})$. Michalek increased the scope of space layout optimization by including terms to minimize heating, cooling, and lighting cost. As in previous studies, he also minimized wasted space or non-living space.

The problem was highly multi-modal and constrained. A first attempt to solve it used CFSQP (a C version of Feasible Sequential Quadratic Programming), but found that while it was fast and robust, the algorithm too quickly converged on a nearest local optima in the multi-modal search space. CFSQP was too dependent on a strong initial starting point. For a better global search, Michalek et al. (2002) tried Simulated Annealing (SA) and Genetic Algorithm. However in the highly constrained field, feasible solutions eluded detection. Ultimately, a hybrid approach of alternating global and local search was used. For a design problem of up to seven rooms, the hybrid SA/SQP approach was able to produce reasonable solutions.

2.5 Fuzzy Multi-Attributive Group Decision-Making

Authors of the studies above commented on the subjectivity of inputs and the need to discern what qualified a good arrangement in order to produce arguably better

solutions. While most of the research above dealt with improving the arrangements algorithms, Ölçer, Tuzcu, and Turan aimed instead to write better inputs and to make a better choice from the Pareto-optimal design alternatives (PODAs) (Ölçer et al. 2006). Their Fuzzy Multi-Attributive Group Decision-Making (FMAGDM) methodology aggregated the opinions of a collection of experts to make the best choice.

A commercial GA-based optimization program, FRONTIER, was used to create a set of Pareto-optimal designs. The N alternatives were each evaluated on K attributes. M number of experts judged each alternative X_j for all its attributes A_i. Their assessments R_{ij} were entered into a three-dimensional array. Ölçer et al. also accounted for weightings for attributes w and experts w_e assigned by a moderator.

Each expert defined a fuzzy membership function for each attribute. Fuzzy methods were used to capture the imprecise nature of their decisions. These functions were normalized across all experts. An actual consensus among experts was not required. All assessments $\mathbf{R}$ were combined to get a net fuzzy opinion. The PODAs were ranked and the best was selected. Selection applied both a closest to utopian and farthest from negative-ideal solution formulation.

2.6 SURFCON

Andrews, Dicks, and Pawling developed an alternate Building Block method for ship arrangement synthesis at the University College London (Andrews and Dicks 1997). The first application of the building block methodology was on submarines in 1990. The SUBCON program (Andrews 1996) and later the SURFCON program (Andrews and Pawling 2003) were developed and ultimately included within the commercial PARAMARINE software. UCL's approach aimed to modernize the traditional ship design spiral sequence: initial sizing, parametric survey, layout, and

performance analysis. Concurrent engineering principles supported a more integrated decision making process (Andrews 2003).

Andrews' approach stemmed from first addressing the ship's functional requirements, and then letting these elements play defining roles in the ship's arrangement and hull form. By delaying the hull form generation step, the hope was to foster innovative hull form solutions, particularly the trimaran.

Only the most important design specifications, such as payload or mission, are considered in the first step. Depending on the size and type of ship, these then drive the major features and topside arrangement in step two. Often the topside arrangement is the battle ground for design decisions that influence the rest of the above and below deck arrangements. Trade-offs in weapon placement and clearance, aviation, and other equipment are dealt with primarily. Initial sizing and weight constraints are then checked for this rough design. The building block methodology is structured to reveal failures early on when they are most easily and cheaply fixed.

After major features, the next breakdown of ship structure is the building block unit. A building block is identified by physical and functional attributes. The blocks fall into one of four functional groups: infrastructure, float, move, and fight. Seven weight groups also characterize the blocks: structure, personnel, ship systems, main propulsion, electrical power, payload, and variable. In three-dimensional space, the building blocks are controlled by drag and drop, and their properties dynamically update as their position is edited.

Functionally related spaces are collocated in Super Building Blocks. Tankage is allocated at this level. When all initial blocks are gathered, a hull form is wrapped around the structure, and external modules analyze weight balance, space utilization, powering, seakeeping and maneuvering, powering, and structural requirements.

Subsequent stages can resolve the building blocks into finer subdivisions. The ship design converges by disaggregating building blocks into spaces.

Ship concepts are validated, but not formally optimized by the building block methodology. The user makes improvements by studying the effect of each block on the design's performance by moving the block and re-evaluating the analysis. Andrews proposed that the goal of preliminary design was specification development, rather than development of the design itself. With the inside-out approach, SURFCON delivers an effective ship concept development tool.

2.7 Ongoing Work

Concurrent in development with the present work is the work of Van Oers et al. from The Netherlands (Van Oers, Stapersma, and Hopman 2007). Their preliminary space allocation software responds to an inferred Catch-22 in the design process. Modern tools give increasingly more advanced output, but require input with a corresponding finer level of detail as well. Thus some preliminary tools are reliant on input that surpasses the preliminary level, and the benefit of the early-stage integrated approach is lost. Van Oers' software, in turn, only takes a rough predefined content list and generic hull envelope and gives a feasible ship concept that is still flexible enough to allow low cost alterations.

Functional space allocation is done on a two-dimensional x-z plane inboard profile grid, with x positive towards the bow and z positive up from the keel. The longitudinal grid unit is 1m. Vertical grid size equals the deck height, 3m. Spaces are all rectangular with fixed dimensions. Space height may exceed one deck. Location may also be fixed at the beginning of the algorithm. An important addition in this work is the explicit representation of required free area around a physical space for proper operation. Having compared relative topology definitions and absolute

position definitions, Van Oers chose the latter "scale coordination parametrization" approach to space location. Van Oers also contrasted the bin-packing versus wrapping approaches, and settled on a hybrid of the two. Available space is enclosed by a generic hull form that could be adjusted slightly to satisfy requirements; the upper envelope is unbounded.

Each space has two independent variables: initial location in x and initial location in z. In sequence, each space is placed, first looking for available area in its initial location. The search continues to the closest horizontal position and then to higher decks until either a feasible location is found or it is returned to its initial location and overlapped with the occupant previously blocking its deposition.

Using the spaces' initial locations as genes, a Genetic Algorithm optimizes the chromosomes of (x,z) coordinates with Tournament selection, Simulated Binary Crossover (SBX) (Deb 2001), and elitism. The objective function has four parts to be minimized: total overlap, center of gravity, total void space, and internal circulation cost. Internal circulation cost is calculated using a similar double summation formulation as seen previously. Overlap is considered between spaces, but also between spaces and the required buffer area around spaces (ie. gun firing arc). Using a population of 150 chromosomes over 150 generations, the total run time takes approximately ten minutes.

Post-processing reduces the population returned by the GA's terminal generation to only designs with zero overlap. After post-processing, the designer is left with a handful of feasible designs from which to differentiate winners. One sample solution is included in Figure 2.2. Van Oers' work shows promising early results, and will eventually be extended to three-dimensions and modified to use a refined space definition and to include passages.

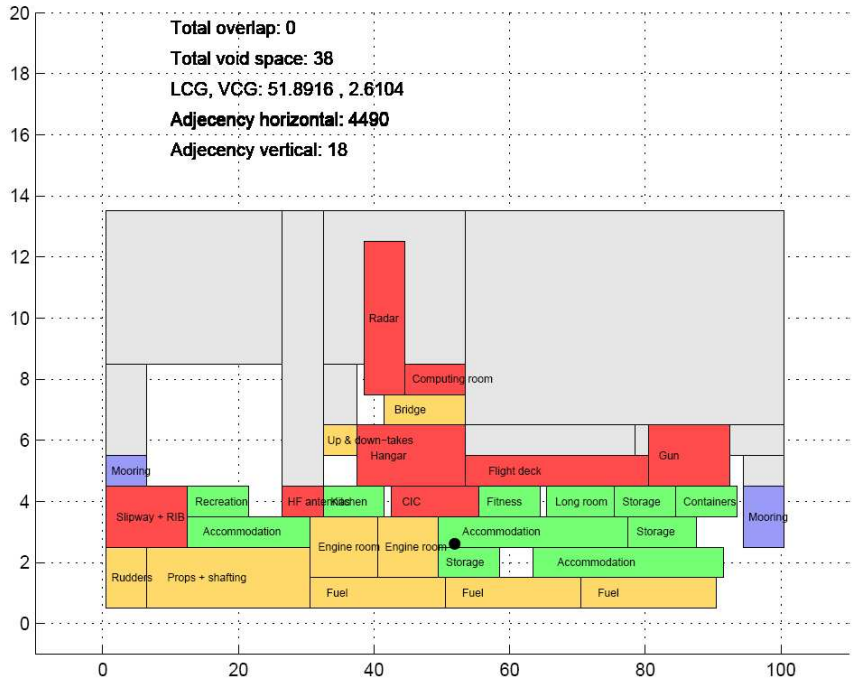


Figure 2.2: A feasible concept design by Van Oers et al. (2007)

2.8 Shared Approaches

A great amount was learned from the past authors. Like Cort and Hills, the algorithm presented here will follow the example of Nehrling by applying Fuzzy Sets to evaluate design criteria. Roughly half of the authors discussed used a Genetic Algorithm as the chosen optimization method. The set-ups of Medjdoub and Yannou and Michalek et al. will be adapted by formulating a two step, topology and geometry, arrangement problem. The two steps will be undertaken after first allocating spaces to Zone-decks. Called a zone in the literature by Cort and Hills, Slapnicar and Grubisic and others, the case for breaking the ship into discrete bins for allocation is well supported from the earlier works. For Zone-decks on the Damage Control Deck, passages will be set as they were by Lee, Han and Roh except with very different controls. All the authors except Andrews and Van Oers used a fixed envelope of the available volume as will be taken as an input here. The assumption

is that the synthesis tool ASSET has sized the envelope adequately. An orthogonal grid overlaying the area is justified by multiple past works. Similar to Michalek's room variable definitions, the geometry solution variable bulkhead locations will be defined with respect to the grid's origin. The three box approach to be introduced is reminiscent of CASGAP's use of chunks. Parallels can also be drawn to work on VLSI (Very-large-scale integration) chip layout. In pursuit of not reinventing the proverbial wheel, the motivation for many design choices made in formulating the presented algorithms has come from these past experiences.

A summary of past and present methods is given in Table 2.1. The first column marks authors who used fuzzy logic to evaluate solutions. GA's were used by five of the included works. Programs limited to manual drag and drop (D&D) methods are marked in the third column. Authors with an X in the "Grid" column delineated the available area with an orthogonal grid. Grid spots or ship regions that were filled in an prescribed order, as by Lee, Han, and Roh, are indicated in the "Fill" column. Most arrangements were made for a known hull form or surrounding boundary, column 6. The included work from the architecture field also used a fixed area envelope. The only Naval Architects who did not use an input hull form were also the two who used Design Building Blocks, DBBs. The incorporation of both allocation (column 8) and arrangement (column 9) of spaces as optimization problems is unique to this author. Those who specifically placed spaces' bulkheads or rooms' walls are credited with performing arrangement. The two step geometry and topology (G&T) arrangement process is indicated in the last column.

Table 2.1: Author and Methods Comparison

	Fuzzy	GA	D&D	Grid	Fill	Hull	DBB	Alloc	Arr	G&T
Carlson, Fireman			X			X		X	X	
Nehrling	X									
Cort and Hills	X		X			X		X		
Ölçer, Tuzcu, Turan	X					X				
Jo and Gero				X	X	X			X	
Lee, Han, Roh		X			X	X			X	
Slapnicar and Grubisic		X		X	X	X			X	
Medjdoub and Yannou				X		X			X	X
Michalek		X		X		X			X	X
Andrews			X				X			
Van Oers		X		X	X		X			
Nick	X	X		X		X		X	X	X

GA = Genetic Algorithm
D&D = Drag & Drop
DBB = Design Building Block
Alloc = Allocation
Arr = Arrangement
G&T = Geometry & Topology

CHAPTER 3

Part 1: Allocation

3.1 Problem Statement

The allocation problem objective is to assign each space i required in the ship to a Zone-deck k with consideration of its global and relative space location preferences, U_{c_i}, and to provide efficient space utilization of the Zone-decks. The available space should be fully utilized, but not over committed. Utilization is the total area of assigned spaces divided by the available assignable area in the Zone-deck. If too much required area is assigned to a given Zone-deck, then each space is compromised by having to shrink in size to fit. Comfort and functionality are lost in these instances. Should too little area be filled by spaces, then each space may be made comfortably larger, but the excess space is ultimately wasteful. Under-utilization should not be as harshly punished as over-utilization. To allow space for unassigned functions such as lockers, minor passages, air shafts, etc, only a significant fraction (perhaps 95%) of the available assignable Zone-deck area would be the goal to be assigned to the spaces.

A schematic profile view of the prototype ship that will be used here as the primary illustration example is shown in Figure 3.1. This four-deck, six-subdivision configuration with seventeen Zone-decks might be typical of a small combatant, such as a corvette. There are two arrangeable decks within the hull and two decks in

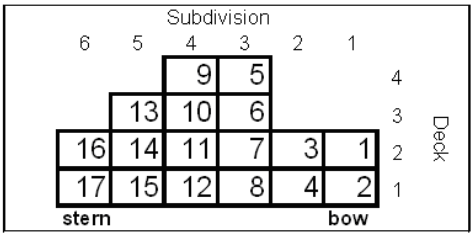


Figure 3.1: Example Ship Inboard Profile

the superstructure. Like the Zone-decks, the spaces are identified by a sequential number.

There are 70 spaces in the example problem to be presented here. To mimic spaces having similar purposes, such as medical, aviation, or accommodations, the seventy spaces are grouped into six groups. All spaces within one group share the same global and relative location preferences, but they do not have preferences to be close to other group members. Each group is color coded to facilitate solution evaluation.

The global Zone-deck location goal is split into separate goals for deck placement and for subdivision placement in this original prototype problem instead of an explicit matrix of preference values for each individual Zone-deck. The total number of discrete global space preferences is the product of the number of spaces with the sum of the number of subdivisions and the number of decks, in this case $70 \times (6+4) = 700$.

Preferences relating the proximity or distance between each pair of spaces are relative preferences. Proximity is measured longitudinally and vertically by increments of subdivisions and decks, respectively. For example, if two spaces are assigned to the same Zone-deck then their proximity is zero in both directions. Discrete adjacency and separation preferences can be stated for up to three subdivisions longitudinally and up to two decks vertically. If a distance between two spaces is greater than these

extents, then the value is taken as the greatest extent stated. For an adjacency requirement, the preference will be high for a near zero proximity and decrease as the two spaces move further apart, either longitudinally or vertically. For a separation requirement, the preference value will be low for a near zero proximity and increase as the two spaces move further apart. Any space can be related to as many other spaces as required by the design.

In the example presented here, there are the 700 individual global location constraints and 850 sets of reciprocal adjacency and separation constraints in addition to the 17 Zone-deck area utilization goals. Each of the 70 spaces can be allocated to uniquely one of the 17 Zone-decks giving a theoretical solution search space of $17^{70} = 1.35 \times 10^{86}$ total possible allocation solutions. After the Zone-deck area utilization constraints are considered, however, many of these solutions are infeasible because each Zone-deck could only accommodate about $70/17 \approx$ four to five spaces if all spaces and all Zone-decks were of average size. The resulting optimization problem is still, however, a large combinatorial problem and quite difficult.

The space allocation problem is an NP-hard combinatorial problem; the solutions are discrete and each solution is unrelated to its neighbors. There is no solution surface with gradient information that can be exploited to help solve the optimization problem. No algorithm has been developed to effectively find a global optimum for the NP-hard class of problems. However, reasonable success has been found using a Genetic Algorithm (Goldberg 1989; Gen and Cheng 1997; Li and Parsons 1998; Li and Parsons 2001; Deb 2001). This problem will be formulated and illustrated for the example ship with 70 spaces and 17 Zone-decks.

3.2 Mathematical Model

3.2.1 Independent Design Variables

The independent variable vector for the allocation problem is defined as an integer-coded chromosome, eq. 3.1.

$$\mathbf{x} = [x_1, x_2, \ldots, x_I] \tag{3.1}$$

I is the total number of spaces and the x_i are integers $[1, 2, \ldots, K]$ that assign space i to one of the K Zone-decks within the ship. This becomes a very difficult combinatorial problem, as noted, since assigning seventy spaces to 17 Zone-decks involves $\mathrm{K}^I = 17^{70} \approx 10^{85}$ possible designs.

3.2.2 Definition of the Goals and Constraints

A Zone-deck is constrained by how much total area can be assigned to it. Each of the K Zone-decks is assumed to have 95% of its assignable area available for spaces. The Zone-deck utilization is defined as the sum of the required areas A_{ik} of all of the spaces I_k currently assigned to Zone-deck k divided by the total assignable area of that Zone-deck A_k, eq. 3.2,

$$\text{Area Utilization of Zonedeck } k \ = U_{U_k} = \frac{\sum_{i=1}^{I} A_{ik}}{A_k} \tag{3.2}$$

3.2.2.1 Zone-deck Utility

The Zone-deck utilization utility U_{Z-d_k} is modeled mathematically as a Normal (Gaussian) distribution function that can take on different standard deviation parameter values for over-utilization and under-utilization, eq. 3.3 and Figure 3.2.

$$U_{Z-d_k} = e^{-\frac{(U_{U_k}-\mu)^2}{2\sigma^2}}, \quad \begin{array}{ll} \sigma = \sigma_{under} & \text{if } U_{U_k} < \mu \\ \sigma = \sigma_{over} & \text{if } U_{U_k} > \mu \end{array} \tag{3.3}$$

This example Zone-deck fuzzy utilization utility is centered at $\mu = 0.95$ to match the assumed assignable area for spaces. For utilization less than or greater than μ, the σs

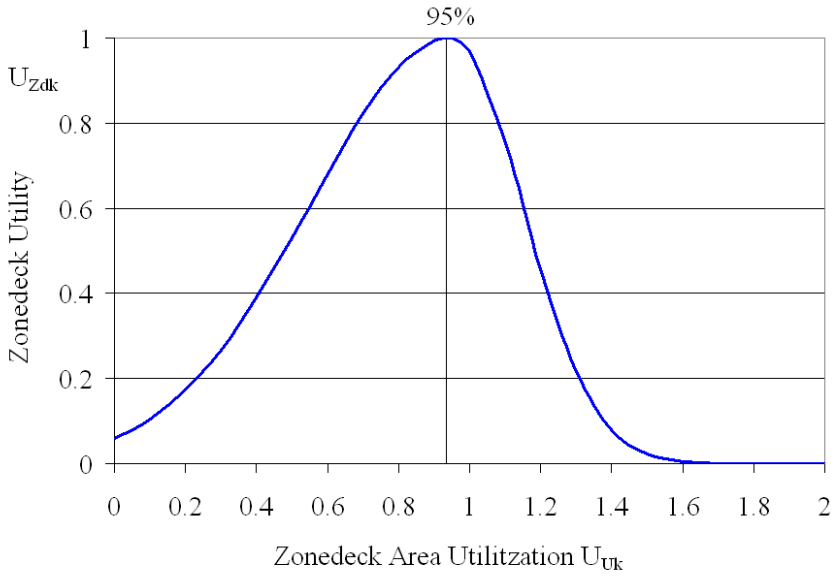


Figure 3.2: Gaussian-Based Model for Zone-deck Area Utilization Utility U_{Zd_k}

shown in Figure 3.2 and used in the example below are $\sigma_{under} = 0.4$ and $\sigma_{over} = 0.2$, respectively. As seen in the figure, all Zone-deck area utilization fuzzy utility values are in the range: [0,1]. Utility $U_{Z-d_k} = 1$ designates full satisfaction (i.e., 95% of the Zone-deck's assignable area is utilized) while $U_{Z-d_k} = 0$ designates no satisfaction or unacceptable. Using minimum correlation fuzzy inference (Kosko 1992), the final Zone-deck utility U_{Z-d}, from the viewpoint of all of the Zone-decks, is the minimum utility of all the 17 Zone-decks, eq. 3.4.

$$\text{Zonedeck Utility} = U_{Z-d} = min(U_{Z-d_1}, U_{Z-d_2}, \ldots, U_{Z-d_{K=17}}) \leq 1 \qquad (3.4)$$

3.2.2.2 Space Global and Relative Location Preferences

The spaces can each have global placement preferences and relative preferences. For simplicity in presenting the example design and to facilitate evaluation of the solutions, color-coded groups of spaces are assumed here to share the same global and relative preferences. Spaces numbered 1-10 (blue), 11-15 (magenta), 16-25 (gray), 46-60 (green), and 61-70 (yellow) are assumed to be related spaces (such as medical,

accommodations, aviation, storeroom/galley/messroom/scullery, etc.). Spaces in the 26-45 (cyan) group have no preference for global location as might occur with storerooms. All fuzzy location utility preferences are discrete to a unique subdivision or deck and between zero and one. For global location constraints, the location of each space in the longitudinal and vertical direction is given a preference value. Table 3.1 shows the global location preference utility values used in the example.

Table 3.1: Global Location Preference Fuzzy Utilities for Space Groups

	Longitudinal Subdivision Preference						Vertical Deck Preference				
	Stern					Bow	Baseline			SS	
Space	6	5	4	3	2	1	1	2	3	4	
1:10	1.0	0.6	0.3	0.2	0.1	0.1	1.0	1.0	0.5	0.4	strong stern and in hull
11:15	0.6	0.8	1.0	1.0	0.8	0.6	1.0	0.6	0.4	0.2	amidships and strong low
16:25	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.5	weak in hull
26:45	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	no preference
46:60	1.0	1.0	1.0	1.0	1.0	1.0	0.2	0.4	0.6	1.0	strong high
61:70	0.1	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0	strong toward bow

Similarly, adjacency constraints between any two spaces are user defined by discrete fuzzy preference values. Adjacency relationships are not required and depend upon the designer's intent. Spaces in the 26:45 (gray) and 61:70 (yellow) groups are assumed to have no relative preferences. All relationships are given twice reciprocally in this example. Table 3.2 enumerates these relationships by space group. For example, each space 1 through 10 has a strong preference for separation from each space 11 through 15 (particularly in the longitudinal direction) and vice versa. The zero location is the current location of space A and the utility values are for the various relative locations of the other space B.

The overall utility of the location of each individual space is also defined using a minimum correlation fuzzy inference returning the minimum preference value of the applicable global position constraints and adjacency constraints for that space.

Table 3.2: Relative Location Preference Fuzzy Utilities for Space Groups

		Longitudinal							Vertical					
		Aft						Forward	Lower				Higher	
Space1	Space2	-3	-2	-1	0	1	2	3	-2	-1	0	1	2	
1:10	11:15	0.7	0.6	0.5	0.1	0.5	0.6	0.7	1.0	0.8	0.6	0.8	1.0	strong sep
1:10	46:60	1.0	0.8	0.7	0.5	0.7	0.8	1.0	1.0	1.0	0.8	1.0	1.0	weak sep
11:15	46:60	0.2	0.3	0.4	1.0	0.4	0.3	0.2	0.3	0.5	1.0	0.5	0.3	strong adj
16:25	46:60	1.0	0.8	0.7	0.5	0.7	0.8	1.0	1.0	1.0	0.8	1.0	1.0	weak sep
26:45														no pref
61:70														no pref

The overall space utility, from the viewpoint of the entire ship arrangement U_{spaces}, is defined as the average of these values, eq. 3.5,

$$U_{spaces} = \frac{1}{\mathrm{I}} \sum_{i=1}^{I} \min(\text{all } U_{global_i}, \text{all } U_{relative_i}) \leq 1 \tag{3.5}$$

where U_{global_i} denotes the vertical and longitudinal global preferences for space i and $U_{relative_i}$ denotes the relative preferences for space i. By taking the average over all spaces, no one space can dominate the overall utility of the solution and prevent an otherwise good solution from being recognized. Within ISA, the global location constraints and the adjacency separation constraints are given as (Zone,Deck) = (x,z) discrete matrices rather than separate values for the Zone (x) and deck (z) as used here. Weighting factors are also introduced to give greater emphasis to the needs of the more important spaces, such as a Combat Information Center.

3.2.3 Definition of the Objective Function

Goodness of the overall arrangement is a function that satisfies both the space location utilities and the Zone-deck area utilization utilities. The goal is to find the maximum U of the overall ship space allocation utility, U^*, by increasing both the Zone-deck utility U_{Z-d}, eq. 3.4, and the overall space utility U_{spaces}, eq. 3.5. Each component utility has a value in the range from zero to one. Fuzzy product inference establishes the overall ship space allocation utility U (Kosko 1992). An

equal weight is given to both component utilities in defining the utility of the overall space allocation utility, eq. 3.6.

$$U^*(\mathbf{x}^*) = \max_x U(\mathbf{x}) = \max_x [U_{Z-d} \cdot U_{spaces}] \leq 1.0 \tag{3.6}$$

3.3 Optimization Method

As introduced previously, a Genetic Algorithm was chosen to solve this challenging optimization problem. The specific GA developed here has evolved from an existing GA (Kent 2005). The Fortran 90 GA code running on a PC with a Pentium 4 CPU, 2.80 GHz processor and 512 MB of RAM can process 4000 generations of this 70 space, 17 Zone-deck example problem in approximately two minutes. A schematic for the Allocation GA is shown in Figure 3.3.

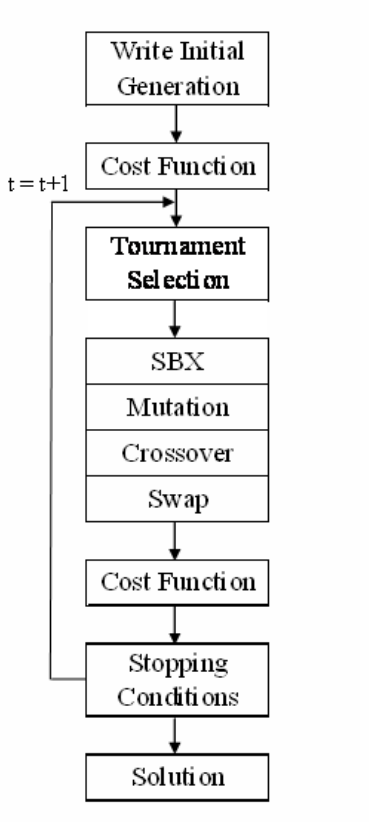


Figure 3.3: Allocation Genetic Algorithm Schematic

Each integer-coded chromosome, eq. 3.1, represents a different candidate alloca-

tion for the spaces within the ship. There is an element or gene for each space to be assigned. Thus, the chromosome's length is equal to the total number of spaces I. The value of the gene, its allele, expresses the numbered Zone-deck assignment for that space. Thus, the integer gene values vary from one to the total number of Zone-decks, K. Fifty chromosomes are included in the population. This number was chosen to give enough population diversity without having too many wasteful extras.

3.3.1 Elitism and Fitness Percentage

The elite chromosome and the fitness percentage (FP) of each chromosome i are identified in the last step of the cost function evaluation. Elitism exempts the elite chromosome from variation operations to avoid replacement by the daughter(s) and mandates its survival in selection to preserve the optimization work accomplished so far. Extra copies of the elite chromosome are frequently present in the population after selection, and they are susceptible to variation. FP is used to rank the chromosomes for consideration in their selection to be re-written into the next generation's population. The FP eq. 3.7 divides the i^{th} individual's fitness U_i, from eq. 3.6, by the sum of all the chromosomes' fitnesses.

$$\mathrm{FP}_i = \frac{\mathrm{U}_i}{\sum_{j=1}^{I} \mathrm{U}_j} \tag{3.7}$$

3.3.2 Tournament Selection

Kent's original selector, a Roulette algorithm, took the selection probability for each chromosome from its fitness percentage. To maintain population diversity, however, a more lenient selection process was introduced. The Double Round Tournament algorithm by Michalewicz decreases the probability that a dominant superchromosome would become over represented (Michalewicz 1996). It is also computationally more efficient and more effective in real-coded problems. Double Round

Tournament selection was used successfully by Li (Li and Parsons 1998; Li and Parsons 2001).

The tournament selection's multi-round process finds each chromosome to be written into the next generation. In the first round, 30 chromosomes of the 50 in the population are randomly selected. This allows for the chance of lesser chromosomes being selected if a super-chromosome and its derivatives were not picked as part of the thirty. These thirty are ordered by fitness percentage and the top three survive to round two. With a random number draw, $0.0 \leq Rnd \leq 1.0$, one of these is selected with a probability corresponding to its fitness rank, eq. 3.8.

$$\text{Survivor Chrom \#} = \begin{cases} \text{Chrom \#1} & \text{if } 0.0 \leq Rnd \leq 0.5 \\ \text{Chrom \#2} & \text{if } 0.5 \leq Rnd \leq 0.8 \\ \text{Chrom \#3} & \text{if } 0.8 \leq Rnd \leq 1.0 \end{cases} \tag{3.8}$$

Thus, the most fit has a 50% chance of selection; the second has a 30% chance, and the third has a 20% chance. Tournament selection is performed 50 times to select each new chromosome in the next generation's initial population.

3.3.3 Variation Operations

Four operations were included to act on the allocation chromosomes. Simulated Binary Crossover (SBX) and two-space swap were added to the two basic operators discussed in chapter one, mutation and single-point crossover, Figure 1.2. Studying the initial allocation results prompted the creation of "swap". This variation operation does not have an obvious biological analogy, but exchanges the values between two genes. For swap to be a viable operation, it must only operate on genes that have the same set of possible alleles. All daughters replace their parents in the population; thus, the elite chromosome is exempt from variation. When called, mutation, single-point crossover, and swap are each executed 25 times. SBX is invoked differently to

control its great global search power.

3.3.3.1 Simulated Binary Crossover

In an effort to increase the diversity in the population and its global searching power, an algorithm by Deb called Simulated Binary Crossover (SBX) was added during algorithm development (Deb 2001). In binary coding, a single crossover point may fall within the bits expressing a variable's value. When segments are exchanged in the crossover, a new value is then introduced to the variable that has been divided. This additional variation is not present in real coding because only values that were present in the parent chromosomes will appear in the daughters. Also greatly increasing its power, SBX is a type of uniform crossover, rather than single or double-point crossover. It operates on all genes in the chromosome during one function call. SBX aims to induce similar variation into real-coded algorithms as present in binary-coded algorithms by formulating daughters as a weighted average of the two parents.

First a random number u is drawn. If u is less than or equal to 0.5, the daughter gene values will lie in between those of the parents. Otherwise the daughter allele will be less than the smaller parent allele and greater than the larger parent allele. The u is used to calculate β, a spread factor, proportional to the difference between the parent genes, eq. 3.9. The difference between the offspring gene values is proportional to the parents' difference.

$$\beta = \begin{cases} (2u)^{1/(\eta+1)} & \text{if } u \leq 0.5 \\ \frac{1}{2(1-u)}^{1/(\eta+1)} & \text{otherwise} \end{cases} \tag{3.9}$$

The spread factor η is a non-negative measure, adjustable by the designer, controlling nearness of the daughters to their parents. A larger η leads to a higher probability of tight solutions. The lowest operable value, $\eta = 1$, is used here to encourage a looser,

or greater, difference from the parents. Daughters x_1^{t+1} and x_2^{t+1} replace parents x_1^t and x_2^t, respectively, using eqs. 3.10 and 3.11,

$$x_1^{t+1} = 0.5[(1+\beta)x_1^t + (1-\beta)x_2^t] \tag{3.10}$$

$$x_2^{t+1} = 0.5[(1-\beta)x_1^t + (1+\beta)x_2^t] \tag{3.11}$$

When the two parents are dissimilar this algorithm is effective. However, it is less effective as they become more similar. This ineffectiveness often occurs in the later generations when many chromosomes share some of the same desirable schema. The power of this operation is maintained by checking to make sure the selected chromosomes are different before engaging SBX. SBX is also treated with a cooling function. The great variety it introduces is less desirable as the population converges in later generations. At each generation, SBX is carried out a "Cool" number of times, eq. 3.12,

$$\text{Cool } = \text{floor}(\text{PopNum}/2 - \text{ floor}(t/T) \times (\text{Comps}/2)) \tag{3.12}$$

where T is the maximum number of generations; t is the generation number. PopNum is the number of chromosomes in the population, and Comps is the total number of spaces. Floor is the round down operator. For PopNum = 50, cool starts at 25 and decreases quadratically to zero in this example.

3.3.3.2 Mutation

Mutation, along with single-point crossover, is a parallel to the process seen in genetics. A single chromosome of the population is chosen randomly by a random real number draw, Rnd, between [0,1] translated into a positive integer chromosome number, eq. 3.13,

$$\text{Parent} = \text{ ceiling}(\text{Rnd} \times \text{Number of spaces}) \tag{3.13}$$

where ceiling is the round up operation. Parallel translation from a real random number to integer selection is used throughout the algorithm. Next, one gene corresponding to a space index number is picked randomly. To select the mutated value from the total number of Zone-decks, all Zone-decks are given an equal chance of assignment with the same random selection pattern. With this process it is possible to select the same value as the parent value. This potential small loss in effectiveness is offset by calling for twenty-five mutations in each generation. With seventy genes each in fifty chromosomes this number of mutations changes only 0.7% of the genes.

Mutation introduces less variation than crossover. Thus, it could be said to be a much less powerful operator than crossover, but it is much more effective in improving the fitness of top chromosomes. Once good chromosomes are found, only small tweaking will find better alternatives.

3.3.3.3 Single-Point Crossover

Single-point crossover exchanges the end tails of two parent chromosomes. Parent chromosomes are selected by random real number draws. Single-point crossover splices the two chromosomes at only one point. The cross point between genes is randomly selected between one and (NumComps - 1) to make sure at least one space is in the head and in the tail. It is an essential operator when the mean and best fitness are relatively poor in earlier generations. In this period, great variation must be introduced to find new regions of the search space.

3.3.3.4 Pair Swapping

Investigations revealed that the best chromosomes only improved in the later part of the search when spaces of like size were exchanged between two Zone-decks (Nick et al. 2006). Neither single-point crossover nor mutation could effectively carry out this trade. Two space swap was introduced to do this further refinement. In

this operation two randomly selected genes of one randomly selected chromosome swap assignments, Figure 3.4. This was found to be more effective than mutation in generating new best chromosomes in the latter part of the searches. This can be seen in Appendix A that tracks the elite chromosome and variation operation that formed it through a complete GA of 3717 generations.

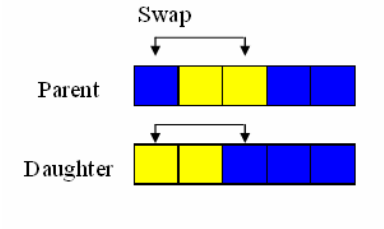


Figure 3.4: Sample Genetic Algorithm Variation Operation: Two-space Swap

3.3.4 Re-seeding and Stopping Conditions

Reseeding and stopping conditions are the only ways to pause and stop, respectfully, the generations. Should the GA not find a new elite chromosome in a user-specified number of generations (e.g. 200) then the progression of generations is paused and a new population is randomly reseeded to replace the stagnant population, less the best chromosome which is retained by elitism. Generations cease when a stopping conditions is reached:

- A maximum number of generations (e.g. 5000)
- A maximum number of reseedings with no additional progress (e.g. 5)

Both conditions imply that the population has probabilistically converged. The nature of a genetic algorithm guarantees a good solution and provides a high probability of finding the global maximum. The ultimate solution is at the discretion of the naval architect after observing the results of a sufficient number of runs.

3.3.5 Observations from Initial Experiments

Most runs were found to converge to significantly different chromosomes. This implies that there are many local optima in the search space. However, the history of the best chromosomes through the optimization runs typically showed convergence through a series of mutations and two-space swaps on a chromosome that was found in the first twenty generations out of thousands. After that, local optima never replaced this strong parent schema. This pattern of convergence results from what is called a super-chromosome. When the majority of chromosomes are poor, a chromosome with moderate fitness will quickly over-populate and become a super-chromosome. Strong schemas have the best probability of surviving through the tournament selection. Copies and altered copies of this original strong parent flood the population. To increase the effectiveness of the GA, the greatest possible diversity in the population should be maintained and it must have the search power to find drastically different chromosomes.

Three specialized features were introduced to address these goals. The first two, tournament selection and Simulated Binary Crossover, were described above. The third modification, GenALLOC, introduced a new way of determining the initial population of the GA to accelerate the progress in the early generations.

3.3.6 GenALLOC

GenSTART was the first initial population generator. In this subroutine each space was sequentially selected and then randomly assigned to a Zone-deck. This led to a large fraction of the chromosomes having very low utility since the area utilization of many Zone-decks was outside reasonable limits. GenALLOC was designed to give better starting points for the Zone-deck utilization by ensuring that in the initial population each Zone-deck has about the same number of spaces assigned to it. This

is achieved by randomly "dealing" out the spaces to the Zone-decks sequentially until all spaces have been allocated. With 17 Zone-decks to be filled with seventy spaces, each Zone-deck is, therefore, randomly assigned either four or five spaces. GenALLOC replaced the predecessor subroutine, GenSTART, and reduced the number of possible solutions from about $K^I = 10^{86}$ to about 10^{75} with each member of the initial population appoximately satisfying the Zone-deck utilization constraints.

3.4 Optimization Results

The best space allocation found for the 70 space, 17 Zone-deck example using 3717 generations of the GA described above is shown in Figure 3.5. For reference, the general space global location and relative adjacency/separation constraints are summarized qualitatively in Table 3.3. The detailed preferences were shown previously in Tables 3.1 and 3.2. The resulting Zone-deck utilities are all 0.993 or above, and from eq. 3.4 the $U_{Z-d} = 0.993$. The individual Zone-deck area utilizations are detailed in Table 3.4. The space utilities average U_{spaces} from eq. 3.5 was 0.7271. A table of the individual space utilities for the final configuration is given in Appendix B. These component utilities give an overall space allocation fitness of $U^* = 0.722$ by eq. 3.6.

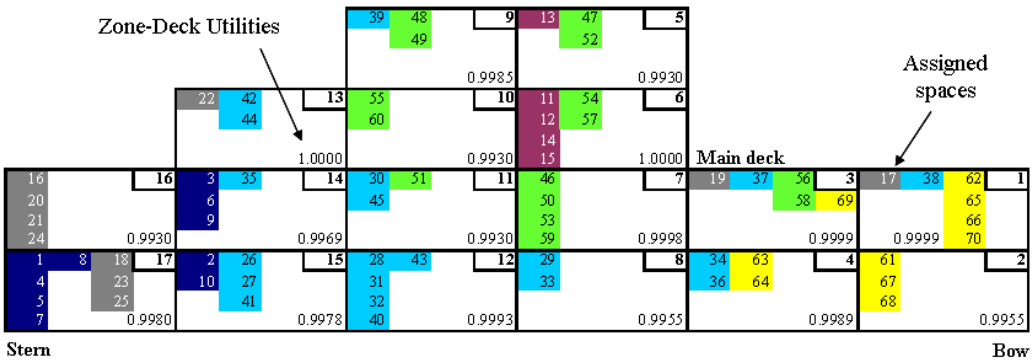


Figure 3.5: Allocation Inboard Profile

Table 3.3: Allocation Space Preference Summary

Global Location Goals		Relative Location Constraints		
1:10	stern, low	1:10	11:15	strong separation
11:15	amidships, low	1:10	46:60	weak separation
16:25	low	11:15	1:10	strong separation
26:45	no preference	11:15	46:60	strong adjacency
46:60	high	16:25	46:60	weak separation
61:70	bow	26:45		no preference
		46:60	16:25	weak separation
		46:60	11:15	strong adjacency
		46:60	1:10	weak separation
		61:70		no preference

Table 3.4: Zone-deck Area Allocation Results

Zone-deck	Area Assigned	Area Available	Utilization U_{U_k}	Utility U_{Z-d_k}
1	267.0	280	0.953	1.000
2	145.4	150	0.969	0.996
3	264.1	280	0.943	1.000
4	232.8	250	0.931	0.999
5	194.8	200	0.974	0.993
6	237.5	250	0.950	1.000
7	207.1	220	0.941	1.000
8	145.4	150	0.969	0.996
9	204.2	220	0.928	0.999
10	116.8	120	0.974	0.993
11	194.8	200	0.974	0.993
12	233.7	250	0.935	0.999
13	142.5	150	0.950	1.000
14	137.8	150	0.918	0.997
15	230.9	250	0.923	0.998
16	194.8	200	0.974	0.993
17	288.8	300	0.963	0.998

While overall the solution shown in Figure 3.5 appears suitable, there are still some spaces that appear to not fully meet the constraints. For example, space 13 prefers to be low in the ship, but it is assigned to the top deck amidships. Seven green spaces are also penalized with a global preference of 0.4 for placement on the Damage Control Deck when they want to be high in the ship. However, the GA does not take this value within the cost function, because their relative preference is even lower at 0.3. Green spaces have a very strong adjacency constraint to the magenta spaces. Space 13, two decks above, controls this constraint causing all five green spaces to have a low score. Thus, the seemingly simple move of 13 to a lower deck would address both issues.

Space 22, in Zone-deck 13, would also be better satisfied at a lower position. However, it is only weakly constrained. Gray spaces are 100% satisfied on both lower decks. On deck 3 they are 0.7 globally satisfied. In Zone-deck 13, their relative preference value is also 0.7 for only being one Zone away from green spaces. This desired separation was also not achieved in Zone-deck 3 where both gray and green are assigned. In fact, for the majority of spaces having a relative preference value less than 1.0, the relative preference value became the governing minimum space utility value.

The individual space utilities for this solution are plotted in Figure 3.6 which shows how many spaces obtained each utility value. Twenty-seven spaces had no global or relative preference. "No preference" is assigned a default value of 1.0 for all locations. These spaces are indicated in the 1.0 utility column. Three spaces have a minimum active global preference of 0.8. Of the 40 spaces that have a relative preference less than one, nearly 75% of those (29) have a relative preference as its minimum preference. These relative preferences ranged from a high of 0.8 to a low

of 0.3. Of the 11 remaining spaces, 10 have equal minimum global and relative preferences and only one, space 13, has a global location value of 0.2 as its active minimum preference.

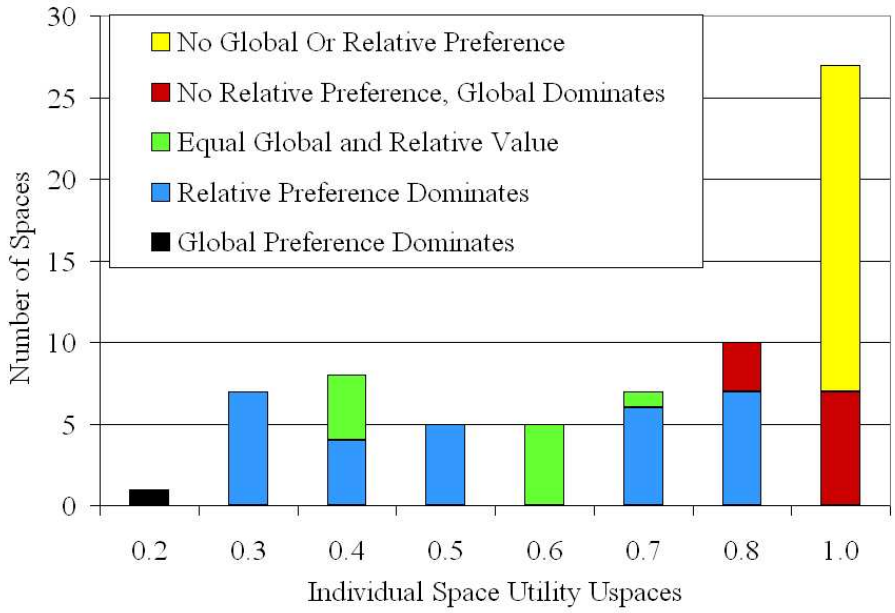


Figure 3.6: Histogram and Analysis of Individual Space Utilities U_{spaces}

For the ship arrangements designer, it is easier to visually identify apparent inconsistencies from among the 700 global preference constraints than it is from the 850 relative adjacency/separation constraints. However, experience has shown that this visually suggested rearrangement rarely proves effective. When the author attempted to improve upon the solution generated by the Genetic Algorithm by visual inspection, only one of 30 proposed rearrangements actually improved the overall utility.

Only proposed changes that maintained at least the same assigned area in each Zone-deck matched or improved the utility level found by the Genetic Algorithm. While Zone-deck utilities are all very high, the algorithm is also very sensitive to this measure with the σs chosen ($\sigma_{under} = 0.4$ and $\sigma_{over} = 0.2$) for the Zone-deck area

utilization constraint. This is a result of taking the minimum over all Zone-decks, rather than the average as is done for the space utilities.

The seemingly obvious rearrangement of space 13 to a lower deck was found to be difficult. Lowering space 13 to deck 2 resulted in the same two deck vertical separation problem with green spaces strongly adjacent to the superstructure. Exchanging 13 with an equal area cyan (no preference) space in Zone-deck 3 gave a lower average space utility (0.7178 as compared to 0.7271). Sacrificing an equal area swap for a placement in Zone-deck 6 with the rest of the magenta group, raised the average space utility from 0.7271 to 0.7414. However, the Zone-deck utility was penalized due to area over-utilization. As a result, the overall allocation utility actually dropped to 0.6898. A dozen other rearrangements attempted for space 13 and space 22 also resulted in poorer overall utility. These efforts were paramount in verifying the importance of the swap operator and the overall effectiveness of the optimization as compared to a manual method.

3.4.1 Parametric Studies

Parametric studies were carried out to compare the three main improvements introduced into the GA for the space allocation problem: GenALLOC, Double Round Tournament selection, and Simulated Binary Crossover (SBX). Figure 3.7 shows the effect on convergence of adding these features successively. Thicker black lines show runs performed using the baseline GenStart and initial Roulette selection. In the next series of tests, GenStart was replaced by GenALLOC generating the red lines. Green lines show the results of replacing Roulette with Double Round Tournament selection. Lastly, the runs colored blue have GenALLOC, Double Round Tournament, and SBX. One blue run was able to surpass the best fitness found without the three additional features. Because each solution starts from a randomly generated initial

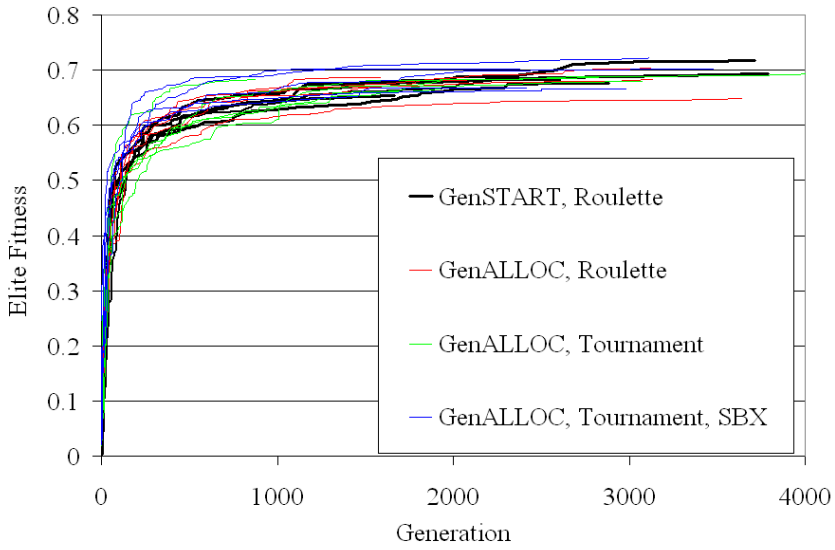


Figure 3.7: Convergence Plots of Allocation Parametric Studies

population, the final solutions vary even when identical algorithm elements are used.

3.4.2 Discussion

Convergence plots of elite fitness versus generation, Figure 3.7, show a steep slope in the early generations. A closer look in this region shows an even quicker convergence rate with the added features. Generally the best fitnesses were found by algorithms that ran longer. This suggests that stopping conditions might be relaxed. The maximum reseedings could be increased from five to perhaps ten or twelve.

Elite fitness in the initial population using GenALLOC is improved up to an order of magnitude above populations written using the purely random method GenSTART. Figure 3.8 gives evidence of this improvement in the early stages of the algorithm. Another benefit of GenALLOC is the effective reduction of the search space. As noted earlier, GenSTART theoretically has about 10^{86} possible solutions for the example problem, while the GenALLOC seeded domain tends to more closely satisfy the Zone-deck allocation constraints effectively leading to about 10^{75} theoretically possible solutions.

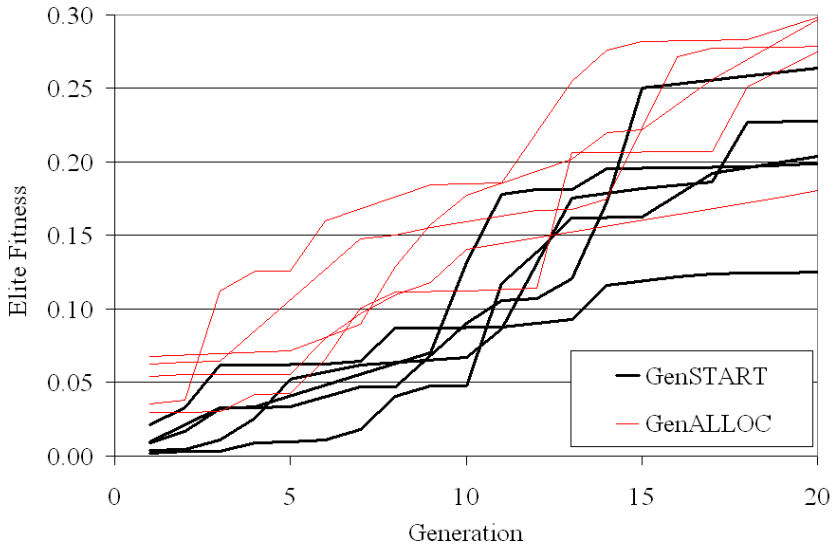


Figure 3.8: Effectiveness of GenALLOC on Early Population Fitness

A study of the generation history of the best chromosomes revealed that after the initial major changes, mutations and swaps provided all of the fitness improvement, Appendix A. This analysis further revealed the extent of population uniformity past the early generations. With this multiplicity of the super-chromosomes, a cooling function is not actually needed on Simulated Binary Crossover (SBX). In fact, perhaps a "heating" function is needed to counteract premature convergence. An alternative would be to reduce the number of chromosomes randomly selected in the first round of the Tournament Selection. The mutation rate could also be increased from acting on 25 genes (of the total $70 \times 50 = 350$) to a much larger number, perhaps 90 ($\sim$25%).

The example space allocation problem limits daughter gene values to integers bounded by one and seventeen, thus limiting the effectiveness of SBX. If values are generated beyond the extents of one and seventeen then the surpassed boundary values are taken instead. Therefore, there is an unwanted propensity to assign values to these two "boundary" Zone-decks. In its defense, upon examining the output

history of best chromosomes of various runs, SBX did cause a greater variation between successive bests. At times, entirely new elite chromosomes were introduced due to this operation. It is also important to note that Zone-deck indices do not strictly correspond to location. Thus, a daughter value significantly different from the parent value does not necessarily signify a new physical location assignment far from the parent Zone-deck assignment.

Since the mean fitness of the population is not of great practical importance, there would be little drawback in maintaining, on average, a less fit and, thus hopefully, a more diverse population. Although the tournament selection parameters used did not significantly produce this intended effect, an even more porous selection method might. This would be particularly critical after a reseeding when the population is most susceptible to being taken over by the super-chromosome held over by elitism.

Greater diversity must be coupled with greater variation. Even more clever operators could be introduced to reach a higher level of fitness. Due to the widely varying space sizes, a one-to-one swap does not typically exchange equal areas, as needed, in and out of a Zone-deck. A dynamic Multi-space Swapping Operator could be used to allow two-for-one, three-for-one, and other combinations of space swaps. Other new operators could have the power to seek out the poorest spaces for relocation in the best chromosome using fitness percentages.

Instead of aiming for more local maxima, the algorithm could be improved in an opposite manner. As later improvements come from variations of the same chromosome, it might be appropriate to switch to Simulated Annealing where, instead of expending computational time on many chromosomes, only one chromosome is manipulated near the end of the solution process.

Along with improving the optimization, the overall utility can also be re-formulated

to better capture the designer's goals. The current overall spaces' utility definition tends to be more sensitive to relative preferences than to global preferences. A weighting factor, γ , between zero and one could be added in the space utility definition to provide more control of these relative influences as shown in eq. 3.14,

$$U_{spaces} = \frac{1}{\mathrm{I}} \sum_{i=1}^{I} \min[\gamma \min(\text{all } U_{global_i}) + (1-\gamma)\min(\text{all } U_{relative_i})]/C_y \qquad (3.14)$$

A larger γ favors global constraints in the minimum selection. The correction C_y would be defined to restore the desirable range [0, 1] for U_{spaces}.

Also, the Zone-deck area utilization utility is highly sensitive. Experience might conclude this over-sensitivity ought to be loosened. The standard deviations of the Gaussian model curve could be increased, or a new utility function defined and evaluated. ISA defines the Zone-deck utility with an added term for the average Zone-deck utilization utility, eq. 3.15 (Parsons et al. 2008),

$$\text{Zonedeck Utility} = U_{Z-d} = min(U_{Z-d_1}, U_{Z-d_2}, \ldots, U_{Z-d_K}) \times \frac{1}{\mathrm{K}} \sum_{k=1}^{K} U_{Z-d_k} \qquad (3.15)$$

An alternative to smarter programming is smarter constraint inputs. Lower values give the preferences more effective weight through the minimum correlation inference. Preference values for the example test problem here were adjusted iteratively using this technique. Spaces with no stated preferences (represented by a preference of 1.0 for all locations) actually may have inhibited other spaces having preferences from finding their ideal location. It is harder to move a space with preference 1.0 from a location than one with a lower preference value. The "No preference" model as used in the example design may not be the most effective model. In an actual design, however, there would be relatively few spaces left completely unconstrained with no preference.

An alternative, hybrid agent-Genetic Algorithm approach has solved this exact problem with a higher utility than found by the Genetic Algorithm described here (Daniels and Parsons 2006; Daniels and Parsons 2007). A hybrid of agents and a genetic algorithm working together has been found to be highly successful reaching a utility level exceeding 0.74 for the test allocation problem presented here. This hybrid approach has been incorporated into the ISA software (Parsons et al. 2008).

CHAPTER 4

Part II: Arrangement

4.1 Two Step Approach: Topology and Geometry

The two step topology and geometry approach for Part II is presented again in Figure 4.1. After the Allocation has been performed, each Zone-deck has known contents, and one Zone-deck is operated on at a time. Topology describes the global and relative positioning of the space centroids within the Zone-deck. This topology is expressed as a real chromosome that is passed to the geometry generation routine. Geometry defines the detailed size and shape of the spaces; the location of the bulkheads defining the spaces. These definitions follow the work of Medjdoub and Yannou (2000) and Michalek, Choudhary, and Papalambros (2002). This double-loop structure embeds the geometry generation routine as the cost function $U(\mathbf{x})$ of the topology Genetic Algorithm optimization. The final arrangement solution for each Zone-deck $U^*(\mathbf{x})$ is thus from the topology that has the best corresponding geometry.

In order of Zone-deck priority, each Zone-deck arrangement is designed and optimized. The default order begins with the middle of the deck (most likely the Damage Control Deck) with the primary longitudinal passages. Major vertical access locations are established on this deck in the optimization. The process proceeds to the lower priority Zone-decks fore and aft on that deck and then below and above. This

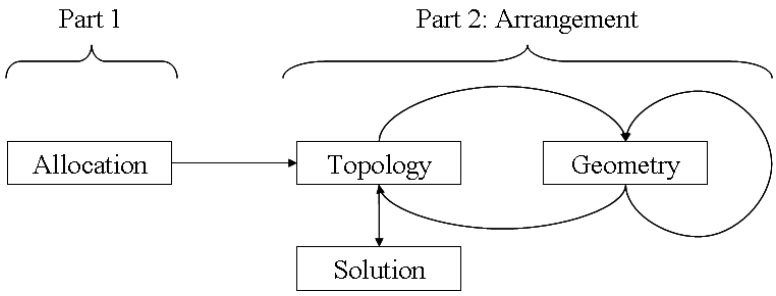


Figure 4.1: Arrangements Optimization schematic

order is editable by the user within ISA.

4.2 Damage Control Deck Zone-decks

Damage Control Deck (DCD) Zone-decks are characterized by having two continuous main passages extending fore to aft along most of the deck. These subdivide the Zone-deck into a port, center, and starboard Sub-Zone-deck (SZD) (Nick and Parsons 2007). In the initial problem formulation within ISA, the designer can edit the default placement of these longitudinal passages, Figure 4.2.

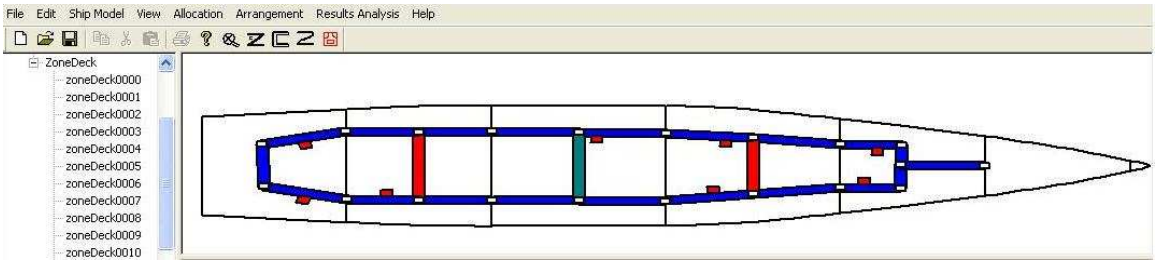


Figure 4.2: Example ISA Damage Control Deck Passages and Stairtowers

Athwartships passages are included here, but not fixed. They are arranged by the topology optimization. Also in this window, the designer designates whether the Zone-deck's stairtowers reside to the inside (in the Center SZD) or to the outside (Port or Starboard SZD) of the main longitudinal passages. Stairtowers have fixed dimensions that can be edited by the designer, but the optimization determines their longitudinal position. Once the stairtowers are located on the Damage Control

Deck they become fixed objects above and below to ensure stair continuity. The stairtower's area is removed from the available area of it's Sub-Zone-deck. Allocation may then be performed to the proper available area for each of the three Sub-Zone-decks.

4.2.1 Topology

Topologies are written preserving the output of the allocation optimization. The topology chromosome has three gene segments for the port, center, and starboard Sub-Zone-decks. The length of each segment is equal to the number of spaces allocated to that Sub-Zone-deck. Segments are concatenated into one integer-coded chromosome delineated by markers for the port (PLP) and starboard (SLP) longitudinal passage locations. An additional arrangeable marker (TP) is included in the center segment if the Zone-deck is designated to have an athwartships passage. The included spaces in each gene segment will stay the same through the optimization as inherited from the allocation, but they will be re-ordered to find the best topology.

The order that spaces are listed in the chromosome represents the longitudinal order (fore to aft) of the spaces. The gene segments are ordered from port to starboard. For example, a DCD Zone-deck with three Sub-Zone-decks, 14 total allocated spaces and an arrangeable athwartships passage might have the following two alternative topology chromosomes.

Topology 1:

Port SZD				Center SZD								Stbd SZD			
1	2	3	PLP	4	5	6	TP	7	8	9	10	SLP	11	12	13	14

Topology 2:

Port SZD				Center SZD									Stbd SZD			
3	1	2	PLP	4	6	9	10	TP	7	8	5	SLP	12	11	14	13

A visual representation of Topology 1 is given in Figure 4.3. The "forward of" and "aft of" relationships are seen as well as the division between the Sub-Zone-decks. The topology chromosome establishes the starting points on a grid used in generating the geometry.

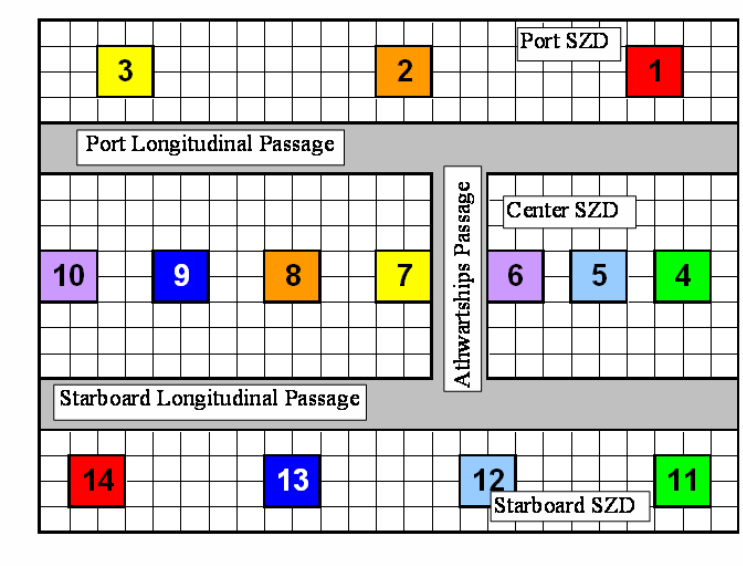


Figure 4.3: DCD Topology Centroids Mapped to the Zone-deck

4.2.2 Geometry

The Zone-deck geometry is modeled on an orthogonal grid. Each grid cell represents equal area at a grid spacing corresponding to the frame spacing, 1 m, or other practical reference distance for the design. From hull form and bulkhead inputs, the dimensions of the available space envelope are known. Longitudinal passages and other fixed objects are set. A curved hull could be accommodated by letting the outer grid spots represent area less than $1m^2$ to match the reduced area available where the grid would be truncated.

The design independent variables used in the stochastic growth algorithm are the placement of the spaces' fore, starboard, aft, and port side bulkheads on the grid. Each space is modeled using a three-box model. Each space is first expressed with only a center (C) box. Up to two additional appendage boxes (A and B) may grow adjacent to the center box. With this three box formulation, spaces can take on the more variable shapes: L's, T's, Z's etc., needed to grow around other spaces, stairtowers, fixed objects, and blockages.

The sample space J shown in Figure 4.4 has grown an additional A appendage box to enable it to become a L-shaped space to fit around the center box of adjacent space J + 1. Space J's corresponding solution variable of bulkhead locations is given in Table 4.1. Location is with respect to the grid's coordinate system origin at the port and aft corner. The four rows are the grid locations for the four bulkheads in the fore ($+x$), starboard ($+y$), aft ($-x$), and port ($-y$) directions, respectively. The three columns of Table 4.1 are for the C center box and for the A and B appendage boxes, respectively. Since there is no second appendage, the third B box column is all zeros in this case. The center box's aft bulkhead and the A appendage's forward bulkhead are at the same location (7) because they must be connected, but there is no physical bulkhead along the overlap. The current geometry solution is always updated graphically on the "Map" (Figure 4.4) and in matrix form (Table 4.1).

Table 4.1: Sample Space J Geometry Variable Matrix

$$\begin{array}{r} \text{+x} \\ \text{+y} \\ \text{-x} \\ \text{-y} \end{array} \begin{bmatrix} 11 & 7 & 0 \\ 14 & 10 & 0 \\ 7 & 5 & 0 \\ 6 & 6 & 0 \end{bmatrix}$$

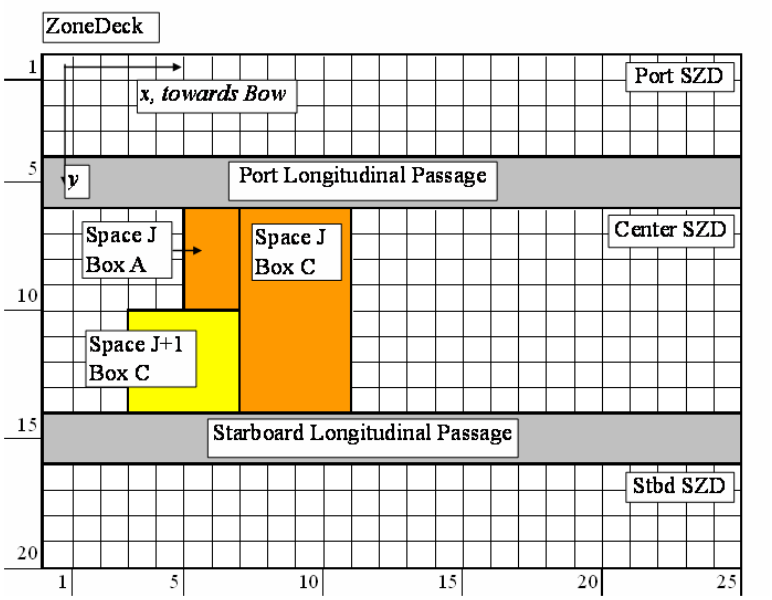


Figure 4.4: Space Representation using Three Boxes

4.2.3 Port and Starboard Sub-Zone-decks

Geometry is created first for the port and starboard SZDs. Due to the relatively thin dimensions of these regions, the Zone-deck length is divided simply by the proportional area requirements of the spaces using the given topology. Thus, there is a bijective relationship from topology to geometry for the port and starboard Sub-Zone-decks. All outer spaces fill the area from hull to passage along their proportional length.

Spaces are purely rectangular except when a fixed object is present or a stairtower has to be placed. In this case, the stairtower area is added to the largest space when assessing proportional area requirements and placing the dividing transverse bulkheads between the spaces. Then the stairtower footprint is taken out of the largest space, which is split into two boxes to accommodate the subtraction. On the port side, the stairtower is placed at the forward extent. On the starboard side, the stairtower shares the aft bulkhead. Of course, the stairtower is placed next to the longitudinal passage, and connectivity for each space to the passage is automatically

assured.

4.2.4 Center Sub-Zone-deck

The center SZD problem is simplified by splitting it into a bow and stern section with the athwartships passage, if one is designated by the chromosome. The sum of the required areas of spaces placed fore of the passage is divided by the total required area. This percentage is multiplied by the Zone-deck length to find the appropriate location to fix the athwartships passage. Only treating one section at a time greatly increases the geometry convergence rate.

After the available area is defined, the center boxes are mapped to the grid. The boxes are first given one grid unit ($1m^2$). Original seed locations are placed in a line longitudinally down the center of the section in order prescribed by the topology. The interval between seed locations is calculated by relative required area. For example, a space in need of a larger area has a larger buffer from other spaces to facilitate its growth into the surrounding area.

The center box's first growth is to the port or starboard longitudinal passage. Once the center box grows to a passage, it "attaches" and is not allowed to shrink from it. In this way, the first access is maintained throughout the entire algorithm by the center box. Based on occupancy, a space may require additional accesses. Up to two access connectivities are evaluated. The two can be required to be on different passages or be indifferent to which passage(s) the accesses are on. A space can also express preference to attach to the port or starboard passage. This is helpful for satisfying an adjacency requirement to another space in the outside SZD or for collecting certain functions on a specific side. If the port or starboard direction has been specified, then the center box extends to the required side. Otherwise, the side is chosen by which one has less required area attached to it already. The center

box also shrinks one grid line away from the center towards its passage to prevent interference with spaces on the opposite side growing fore and aft.

The last initial expansion of the center boxes is in the fore and aft direction. In the bow and in the stern section, the spaces that are forward and aft are allowed to grow longitudinally to that extent. This expansion is also fixed so that the space may not shrink from this direction. If there is only one space in the section, then it automatically fills the entire available area. Figure 4.5 shows a center Sub-Zone-deck after the center boxes have undergone their initial expansion.

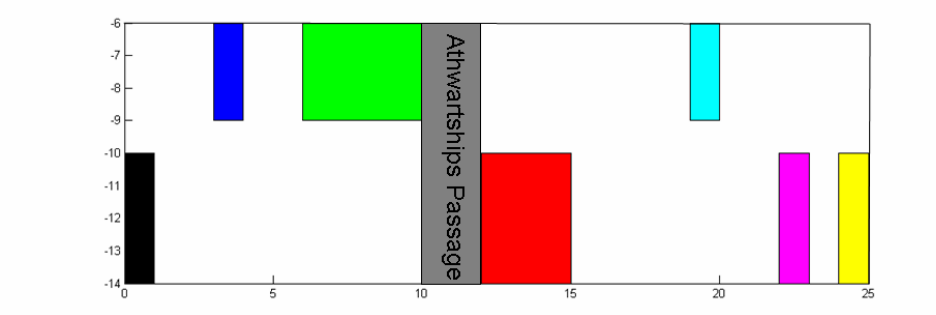


Figure 4.5: Center Box Initial Expansion in the Center Sub-Zone-deck

To give the spaces on the Damage Control Deck the highest priority to satisfy their needs, the stairtowers, if applicable, are first placed amid the initial center boxes. The initial placement is found in two steps. First the bow or stern section is chosen. If both the port and starboard stairtowers were set to the inside of the passages then the port stairtower is placed in the bow and the starboard stairtower is placed in the stern. This split is made to assure a minimum separation distance. If only one of the stairtowers was set to the inside, it is placed in the section that has more leftover area (or less over committed area) from its known available area and total spaces' required area. If the stairtower examines a section that has already been filled by a single space, then the single space can be split into two boxes to accommodate the stairtower similar to what is done with the Port and Starboard

Sub-Zone-deck's largest space. A port stairtower searches within its section from fore to aft for available open length. A starboard stairtower looks from aft to forward for a place where it can fit between the center boxes next to the starboard longitudinal passage. The stairtower is not, however, fixed in this initial location. During the stochastic growth loop, an adjacent space can push the stairtower longitudinally as it attempts to grow.

4.3 Stochastic Growth Loop

Once the center boxes have been expanded and stairtowers have been added as appropriate, the stochastic growth loop is activated. The basic loop sequence of this algorithm is shown in Figure 4.6. Each iteration begins with three selections to determine the attempted move: Space, Direction, and Growth. Then the move is tested according to a series of rules. If the move is allowed, the Map and the space's bulkhead locations in the independent variable matrix are updated. Looping continues until a stopping condition is reached. The converged arrangement is then evaluated.

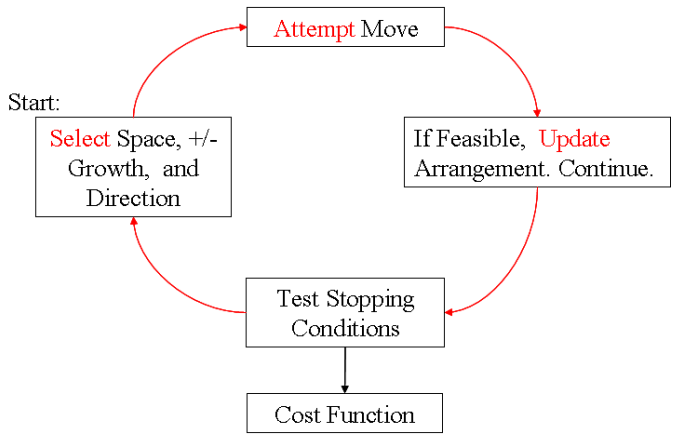


Figure 4.6: Stochastic Growth Loop Schematic

The cost function reveals strong solutions obtained by applying a method of reasoned randomness to the initial selections. Each of the three selections are made

probabilistically to improve the arrangement. For example, a small space has a higher probability to grow rather than to shrink. However, the random draw may return a move that is disadvantageous. This is necessary to prevent pre-mature convergence or stagnation. Even if one space is fully satisfied, it may have to move in an unfavorable way to allow another space to attain its needs.

4.3.1 Probabilistic Selections

First, a space is selected. The probability of selection of compartment j, $Psel_j$, is based on space area needs and the remaining unoccupied area using eq. 4.1. AA_j is the current amount of acquired area; RA_j is the area required for the space. NumSp stands for the number of spaces in the SZD section (bow or stern). SZDarea is the available area of the section, and Open is the area currently left unoccupied in the section. NoFix is a nondimensional multiplier.

$$Psel_j = \frac{|AA_j - RA_j| + \text{NoFix} \times \text{Open/SZDarea}}{(\sum_{i=1}^{NumSp} |AA_i - RA_i|) + \text{NumSp} \times \text{NoFix} \times \text{Open/SZDarea}} \tag{4.1}$$

A larger difference between required area (RA) and acquired area (AA) leads to a higher probability of that space's selection. Due to the second term in the numerator, however, there is always a non-zero chance of selection. Therefore, no compartment becomes pre-maturely fixed before all available space is occupied. The NoFix multiplier is used to give an appropriate weight to this second term. In this formulation it has the numeric value of the length of the SZD currently being arranged by the stochastic growth loop. Without the multiplier the effect of this term is negligible. If the multiplier is too large then the selection is too random rather than being based on need expressed by the first term. Selection probabilities are normalized to fall between zero and one. A Genetic Algorithm style Roulette Selection method is applied to choose a space from these probabilities.

Second, the growth value is chosen based on the selected space's area needs. Area Satisfaction, AS_i, is calculated for the selected space i, eq. 4.2.

$$AS_i = \frac{AA_i - RA_i}{RA_i} \tag{4.2}$$

A negative AS indicates a need to grow, and thus, the space is given a higher probability to grow and a lower probability to shrink. Six regions of Area Satisfaction are defined as: AS $\leq$ -0.5, -0.5 $<$ AS $\leq$ -0.25, -0.25 $<$ AS $\leq$ 0, 0 $<$ AS $\leq$ 0.25, 0.25 $<$ AS $\leq$ 0.5, and AS $>$ 0.5. Each region has unique probabilities for positive or negative growth (shrinkage). Probabilities are derived from a probability density function, eq. 4.3, with mean $\mu = 1$ and a standard deviation σ that can be edited by the designer. The x values for eq. 4.3 are found in Table 4.2 for each region.

$$\text{PDF}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.3}$$

Table 4.2: Probabilities for Space Growth Values -3 to +3

Growth	AS $\leq$ -0.5	-0.5 $<$ AS $\leq$ -0.25	-0.25 $<$ AS $\leq$ 0
-3	$\mu + 5\sigma$	$\mu + 4\sigma$	$\mu + 3\sigma$
-2	$\mu + 4\sigma$	$\mu + 3\sigma$	$\mu + 2\sigma$
-1	$\mu + 3\sigma$	$\mu + 2\sigma$	$\mu + \sigma$
1	$\mu + 2\sigma$	$\mu + \sigma$	μ
2	$\mu + \sigma$	μ	$\mu + \sigma$
3	μ	$\mu + \sigma$	$\mu + 2\sigma$
Growth	0 $<$ AS $\leq$ 0.25	0.25 $<$ AS $\leq$ 0.5	AS $>$ 0.5
-3	$\mu + 2\sigma$	$\mu + \sigma$	μ
-2	$\mu + \sigma$	μ	$\mu + \sigma$
-1	μ	$\mu + \sigma$	$\mu + 2\sigma$
1	$\mu + \sigma$	$\mu + 2\sigma$	$\mu + 3\sigma$
2	$\mu + 2\sigma$	$\mu + 3\sigma$	$\mu + 4\sigma$
3	$\mu + 3\sigma$	$\mu + 4\sigma$	$\mu + 5\sigma$

The far left column in Table 4.2 indicates the growth value in grid spacing units. A space may have a positive or negative growth value up to three. There is no option

for a growth value of zero. In the column under each region, values are taken from the Normal PDF at the indicated x variable values to find the relative probability of selecting the corresponding growth value in the left hand column. For example with σ=0.2 and AS = 0.1, a growth of -1 will have the probability corresponding to PDF(μ) = PDF(1) = 1.99. The probability of growing by +2 is PDF(μ + 2σ) = PDF(1.4) = 0.27. These PDF values are normalized by the total sum of its column. Again, a Roulette Selection method picks a growth value from a random draw between zero and one.

A positive growth value will cause the selected space to try to grow larger by one grid unit at a time, and multiple moves up to the space's growth value can be made in a single iteration. Particularly at early iterations when the spaces are all small and somewhat spread out, this stepping mechanism allows for quicker area filling. The three selection functions constitute a substantial component of the computational time of each iteration. By allowing multiple moves without re-selecting the move, efficiency is increased.

If a growth attempt fails, the remaining moves are not attempted as they would also fail. Failure here is defined as encountering any blockage. Both growth and shrinkage have additional rules associated with them as described later on. Logically, a negative growth value will prompt shrinkage. One grid unit is attempted per iteration. Multiple shrink steps are not taken regardless of a growth value $<$ -1.

The last selection made is to one of four directions. This choice is based on aspect ratio. The overall dimensions Δx and Δy are found by examining all boxes' bulkhead locations, where x is the longitudinal direction and y is the transverse direction. If a compartment is to grow, it will have a tendency to select a direction that favors increasing its shorter dimension. Similarly, a shrinking compartment is more likely

to shrink from its longer dimension. If the aspect ratio is less than 0.5, then the probability to increase aspect ratio is even greater. These probabilities expressed as a percentage are all found in Table 4.3. The direction is selected by a random number draw correlated to one of the four directions by the Roulette Selection method.

Table 4.3: Percentage Probabilities for Grow and Shrink Growth Directions by Aspect Ratio

	Grow				
	$\Delta x < .5\Delta y$	$\Delta x < \Delta y$	$\Delta y < .5\Delta x$	$\Delta y < \Delta x$	$\Delta y = \Delta x$
$Psel_{+x}$	40	30	10	20	25
$Psel_{+y}$	10	20	40	30	25
$Psel_{-x}$	40	30	10	20	25
$Psel_{-y}$	10	20	40	30	25
	Shrink				
	$\Delta x < .5\Delta y$	$\Delta x < \Delta y$	$\Delta y < .5\Delta x$	$\Delta y < \Delta x$	$\Delta y = \Delta x$
$Psel_{+x}$	10	20	40	30	25
$Psel_{+y}$	40	30	10	20	25
$Psel_{-x}$	10	20	40	30	25
$Psel_{-y}$	40	30	10	20	25

4.3.2 Move Rules

Before a move is attempted, the algorithm identifies the selected space's boxes that are allowed to move in the selected direction. If any appendages have already grown in that direction, then these appendages may grow and the center box will not. Otherwise, with no appendages in front of it, the center box is the first active box. Next, the algorithm looks for appendages in the directions perpendicular to the selected direction. These appendages attempt the move as well. An appendage in the opposite direction from the selected direction never moves. There is always at least one box active, and all three boxes may be active. Each active box takes a turn attempting the move.

A valid move must be within bounds. Bounds are defined by passages and watertight bulkheads in the Center Sub-Zone-deck and by the hull and watertight bulkheads on Zone-decks below the Damage Control Deck. If the selected space has

positive growth, the move fails if it tries to grow past the boundary.

A successful growth move can only be made into available grid spots. No two compartments (space, passage, vertical access, or any other blockage) may occupy that same spot at the same time. Therefore, if growth is blocked anywhere along the length, then the move fails or additional rules come into play depending on what the blockage is and whether a center box or an appendage box is active. If all adjacent spots are open, then the bulkhead is moved.

Each adjacent spot along the length of the bulkhead attempting to move forward is checked one by one. Unoccupied regions along the length covering at least the minimum dimension are called FreeSpots. All FreeSpots and in turn, blocked regions are found. If a single FreeSpot encompasses the entire length, then the box is unblocked and it grows. The move simply fails if an appendage is blocked at any spot.

If a center box is blocked by another space or something immovable, then it will grow an appendage into a FreeSpot. Multiple appendages can grow adhering to the rule of no more than two total appendages. If there is more than one FreeSpot, the one that shares a side bulkhead location with the center box is selected. This helps maintain regular shapes. If no FreeSpot is available at the beginning or end of the searched length, then the choice is made randomly. Once the center box is blocked, the remaining growth steps to the growth value are not taken.

Special blockage situations arise when a box runs into a moveable object like a stairtower on the Damage Control Deck. This may be pushed fore and aft along the passage. A space growing transversely clearly can not push the stairtower. Same as other moves, first it must be verified that the stairtower will not be pushed outside the extents, and second, the spots on the far side of the stairtower must be open.

If both of these conditions are met, the stairtower is pushed. Conversely, on the below Damage Control Deck Zone-decks, the stairtower is fixed and the passages have variable length longitudinally.

A successful growth move is tested to see whether the selected space's bulkhead has reached an extent. Once a box reaches an extent, it is advantageous to adhere to it. This reduces chattering and speeds convergence. The "NoShrink" test examines the space's current area and aspect ratio. If the space's acquired area exceeds required area by more than 10%, then the growth does not stick. When it is already too large, it is undesirable to prevent shrinking. The active box must also have an aspect ratio within the acceptable limits. It is important to make sure it is a good placement before fixing a bulkhead.

Each box individually records in which directions' bulkheads are fixed, or "NoShrink" directions. These bulkheads cannot grow larger without violating the border constraint and they are not allowed to shrink either. Initial center box expansions that attach to passage and the fore and aft extents are recorded as NoShrink directions. When a negative growth value is chosen, the move first queries whether the selected direction is a NoShrink direction. If so, the shrink move fails, and the looping continues.

Negative growth moves have less requirements than positive growth. A center box is not allowed to shrink smaller than a specified minimum dimension. An appendage that attempts to shrink smaller than the minimum dimension in either the x or y direction is simply deleted. Before a center box shrinks, the program checks to see if it would disconnect from any appendages on the perpendicular sides from the shrinking bulkhead. If the overlap between center box and appendage decreases too much then the appendage is deleted. Similarly, when an appendage attempts

to shrink and it decreases the overlap such that the center and appendage box are essentially two separate spaces, then the appendage is deleted. Shrink rules maintain required connectivity between a space and its passage and between appendages to the center box.

4.3.3 Appendage Repair Functions

Additional repair functions preserve the integrity of the three-box formulation. The first two functions look at redundancy. An appendage that is as wide as the center box no longer looks like a separate appendage. In this case, it is included in the center box. For example, an appendage box in the $+x$ direction sharing the same $-y$ and $+y$ bulkhead locations as the center box is deleted and the center box's $+x$ bulkhead is re-written in the position of the deleted appendage's $+x$ bulkhead. One rectangle remains from the two abutting rectangles. In the same way, two adjoining appendages that grew in the same direction and to the same length away from the center box are re-written as one appendage. These two repair operations are called for every space every ten iterations.

It is also beneficial to periodically delete small appendages. Before this clean-up repair, final solutions often had small jogs in the spaces' overall outer footprint due to appendages that were only 1 unit long. Although they are deleted when they shrink past minimum dimensions on any iteration, appendages only start growth one unit long. Every 50 iterations, appendages that are still one unit long are deleted. The flexibility to have two appendages at once is essential while they are continually grown and deleted in the stochastic growth loop even though the best solutions rarely contain spaces with two appendages.

4.3.4 Looping Rules and Stopping Conditions

One iteration of the stochastic growth loop may attempt the same move for multiple boxes of the selected space and for multiple steps. After a failed move, the next box attempts the move. Once all boxes have tried to move, the move is repeated if the growth value is greater than one. After the steps finish, the next iteration begins with the three selection functions.

A tally keeps track of how many grid spots are left open at each iteration. Looping ceases when one of two stopping conditions is reached. Either a maximum number of iterations has been reached or there are no more open spots left. Smaller sections arranged on the Damage Control Deck typically exit upon filling all the available area. Below, the the maximum number of iterations is usually the active stopping mechanism.

4.4 Below Damage Control Deck Zone-decks

4.4.1 Stairtowers and Passages From Above

Zone-decks below the Damage Control Deck (DCD) are arranged after the Zone-deck above it on the DCD is arranged. The stairtower on the DCD is continued down to the lower decks and fixed. Passages are drawn adjacent to the stairtowers to the inside or outside as they were above.

A variety of stairtower and passage configurations from the DCD must be handled on the lower decks. To reduce the number of configuration cases explicitly treated, the two stairtowers are simply compared to find which one is fore and aft. If neither is ahead of the other, the port stair-tower is labeled fore. The two cases (Port forward or Starboard forward) proved to be robust enough to handle all configurations.

Unlike on the DCD, the longitudinal passages do not have to extend the length of the Zone-deck. The watertight bulkheads are not penetrated. Instead, passages

start only one unit longer than the stairtowers. The forward passage extends one unit towards the aft, and the aft one goes forward. The spaces control how the passages grow and shrink. Therefore, there is no internal division to define multiple Sub-Zone-decks. The entire available area in the lower decks is considered one Sub-Zone-deck and arranged all at once by the stochastic growth loop.

4.4.2 Topology

The below DCD topology chromosome is not split by indicators for passages; it contains only space index numbers. Space order in the chromosome only roughly translates to a port to starboard and a fore to aft placement as compared to the DCD chromosome. Rather than follow a strict definition of topology, chromosome order is mapped to the center box attachment positions along the two passages.

There are essentially two kinds of attachments: a side and an end attachment. Side attachments are to the port or starboard side of the passage. A center box's side attachment is a NoShrink direction (it can neither grow nor shrink in that direction). End attachments are not NoShrink directions to allow for the passages' variable length as the end space grows and shrinks in the attachment direction. If two attachments are required, they will be either one side and one end attachment or two side attachments. Center boxes are not placed with two end attachments due to the greater control an end attachment has over the passage length.

The number of access attachments is primarily determined by the space's personnel capacity, particularly in battle manning. This center box placement routine can accommodate zero and up to as many as all of the spaces having two connections. Although this latter scenario is rare and results in poor arrangements; the algorithm left more available area unoccupied particularly to the outside of the passages. Better arrangements are found with fewer constraints for multiple connectivities.

The chromosome follows a transverse order like on the DCD by placing the first spaces on the port passage and the last spaces on the starboard passage. This, like many other design choices, is done based upon the spaces' area requirements. Half of the total area required in the Zone-deck is calculated. In order as laid out by the topology chromosome, spaces are assigned to the port passage while summing the accumulated required area. When the accumulated area surpasses half of the total required area, the last assigned space is instead placed on the starboard passage if half of the number of spaces have already been added to the port passage. If less than half of the number of spaces have been assigned to the port passage, then it stays with the port passage. The remaining spaces are assigned to the starboard passage. Each space keeps its chromosome order relative to the other spaces on its passage to assign longitudinal position.

The smallest space, and only the smallest, is relegated to the outside of the passage unless it requires two connectivities to access. If its capacity demands two accessways then no space is placed to the outside, and the smallest is treated based upon its longitudinal order dictated by the chromosome. This logic was added after generating many geometries and finding that the best geometries consistently had this layout. All other spaces attach to the ends or to the inside of the passage.

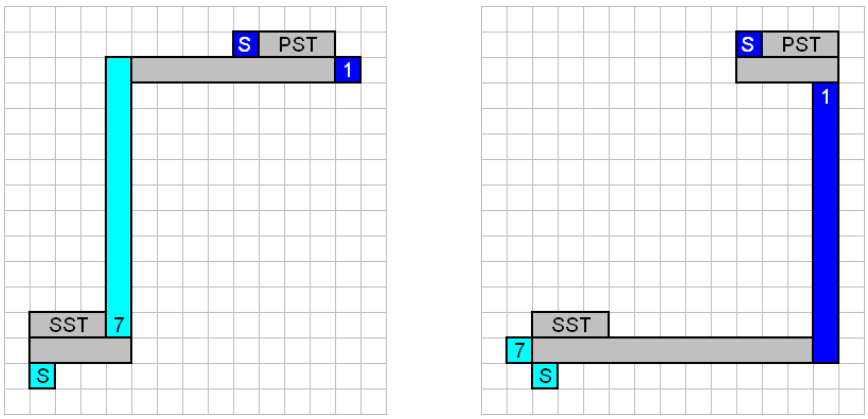
Remaining spaces are placed in six steps. The attachment routine will be presented here for only the port stairtower (PST) forward case, but the steps are simply mirrored for the opposite case. In the following figures, four additional spaces are assumed to have been assigned to the port passage (shown in blue), and with three additional cyan spaces assigned to starboard. The numbers one to seven correspond to the space order in the chromosome and S is the smallest space already placed to the outside. These three graphics, Figures 4.7 - 4.9, show only the relevant portion

of a Zone-deck to illustrate the remaining spaces' placement on the passages by the following six steps:

1. Most forward space on forward stairtower
2. Most aft space on aft stairtower
3. Spaces along the inside length of the forward stairtower
4. Spaces along the inside length of the aft stairtower
5. Most aft space on the forward stairtower
6. Most forward space on the aft stairtower

In step one, the first space on the forward stairtower is placed (unless the first was the smallest, and then the second is used). It is positioned at the fore end of the passage sharing the passage's forward bulkhead with its center box's aft bulkhead. However, if the stairtower was placed at the forward watertight bulkhead, then this placement is not available. The space is brought to the inside at the forward extent of the passage. Should this space require two access points, then it affixes to the port passage's side and grows transversely to the starboard passage which grows forward to meet it. The starboard passage fore end is capped by the double access space. Step two repeats this logic, except for the most aft space of the aft passage. A single attachment in step one and a double attachment in step two is shown in Figure 4.7(a), while the opposite case is illustrated in Figure 4.7(b).

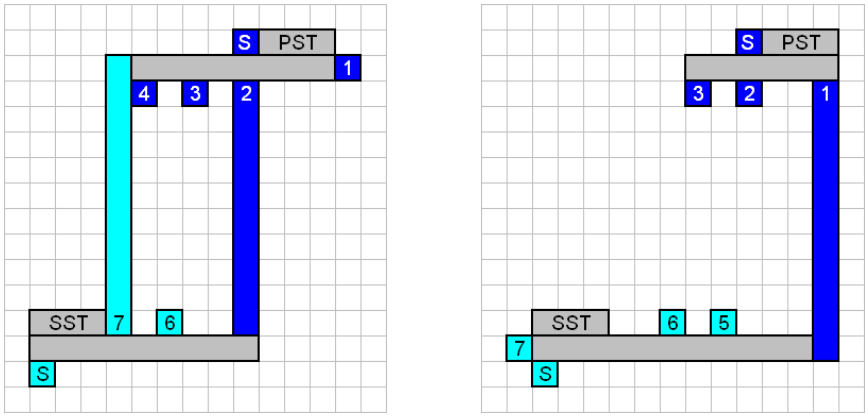
The next step returns to the forward passage to place spaces along its inside length. If the aft starboard space did not grow over to cap the end of the port passage in step two, then the last port space is left to the fifth step. Otherwise it is included here; see space 4 in Figure 4.8(a). Progressing fore to aft for the remaining spaces in



(a) (b)

Figure 4.7: Center Box Attachments Example Steps 1 and 2

the order dictated by the chromosome, the center boxes are put on the inside of the passage starting just aft of the stairtower. Each subsequent center box is placed two units further aft than the last one growing the passage aft along as needed, ie. space 3 in Figure 4.8(b). If any of these spaces require two passage attachments, then it grows towards the starboard, and the starboard passage grows forward to meet it (if the passage did not already grow forward in step one, or for an earlier center box in this step). Space 2 in Figure 4.8(a) demonstrates this behavior. The same procedure is followed in step 4 to place spaces along the inner length of the starboard passage progressing aft to fore.



(a) (b)

Figure 4.8: Center Box Attachments Example Steps 3 and 4

At step 5, only two spaces maximum are left to cap the passages. In the (a) example, space 4 was included in step 2, but in the B example it is added in step 5 at the aft end of the port passage. Step 6 attaches the most forward space on the starboard passage to its forward end. Space 5 in Figure 4.9(a) completes the routine in this way. Should the addition of a double access space in step 6 require that the port passage grow aft to meet it, then the space added in step 5 is re-drawn at the new position on the longer passage. This is the only repair needed when following the outlined steps.

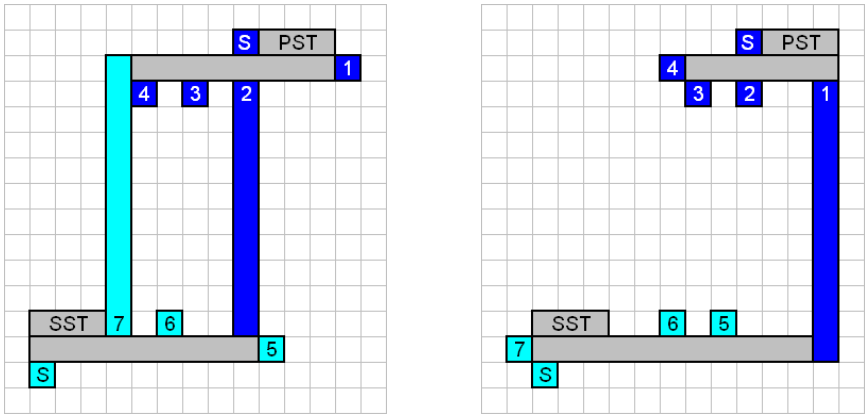


Figure 4.9: Center Box Attachments Example Steps 5 and 6

The routine's specific order prevents interference between the center boxes and satisfies the need for access while also preserving the transverse and longitudinal topology given in the chromosome. Passages grow to keep the center boxes spread out and to allow for spaces to reach between the passages for double access, if required.

Stairtowers that roughly trisect the Zone-deck length generate the best arrangements. Stairtowers placed very close together on the DCD leave limited room for space attachment and growth. When the stairtowers are separated by a longer distance, the passages must grow longer using up valuable area. It also leaves little room between the stairtower and the watertight bulkheads for end cap spaces to

grow. Logic within the DCD arrangement algorithm helps to produce this preferred stairtower spread distance.

End attachments are made when possible. It was found that leaving these open (in lieu of making more side attachments) resulted in the nearby grid spots remaining unoccupied. Some topologies obviously will result in better geometries, but it is left to the optimization to reveal the best.

The stochastic growth loop is called after the center boxes are mapped. The same move rules apply to the below DCD Zone-decks supplemented by new rules for passage growth. As before, once one of the two stopping conditions is reached, the loop exits and the converged arrangement is evaluated.

4.4.3 Passage Growth Control

Side and end attachments can both grow and shrink the passage(s) to which they are attached. If the space is attached to two passages, then it will attempt to adjust both passages as necessary. Only center boxes have this control as they grow and shrink in the fore and aft directions. Passages only orient longitudinally. Transverse passages were not found necessary. Passages should only be as long as needed to reach all the spaces attached to it. It is preferable to assign available area to the spaces rather than to circulation.

Passage growth is activated by a space attempting to shrink away from the passage, such that the passage must grow to stay connected. An end space shrinking from its attachment direction will automatically "pull" the passage. Since the passage grows into a spot freed by the shrinking space no additional test is necessary to allow the move. Each time a space attached on a side shrinks from the fore or aft direction it tests to make sure that it maintains the required overlap for access. If, for example, a shrink from the aft direction would bring the aft bulkhead to the

same location as the passage's fore bulkhead (zero overlap), then the space tries to pull the passage forward. The grid spot just beyond the passage must be open to let the passage grow. Often, the end is capped, and the pull fails. If the side space cannot pull the passage to retain access, then the space's shrink move fails.

Passages shrink when a center box grows. This typically occurs after the adjoining space has shrunk away (pulling the passage) and it is now growing back. Two checks are performed. First, the passage is not allowed to shrink if it would disconnect from any other spaces. Second, the passage will not shrink shorter than its stairtower's extents. Any time an end attachment grows in the direction of the passage it has to perform these checks to allow the growth and passage push. The side attachments only check when their previous attachment position may have been the one limiting the minimum passage length and the growth might enable the passage to be pushed back. Passage are only as long as is required to keep contact.

4.5 Mathematical Model

4.5.1 Definition of the Goals and Constraints

Geometries are evaluated on eight criteria. Five criteria are for shape: required area satisfaction RAS, aspect ratio AR, minimum overall dimension MOD, minimum segment dimension MSD, and perimeter length. Adjacency and separation preferences to other spaces are examined, as well as the connectivity requirements. Criterion success is assessed using fuzzy utility membership functions. Once the measure of interest x is found, it becomes the argument of a fuzzy preference function y= $U(x)$ to calculate a corresponding fuzzy utility value between zero and one. Table 4.4 shows the control points used to define a generic linear piecewise continuous fuzzy utility versus criteria measure, x. B and t stand for bottom and top, respectively. L and u stand for lower and upper, respectively. The resulting utility

function is shown in Figure 4.10. These utilities are obtained for each space from the default space constraint database and then edited, as required, by the designer to reflect unique requirements for the design.

Table 4.4: Control Points for a Piecewise Linear Fuzzy Utility

x	0	bl	tl	t	tu	bu	b
y= $U(x)$	0.0	0.05	0.95	1.0	0.95	0.05	0

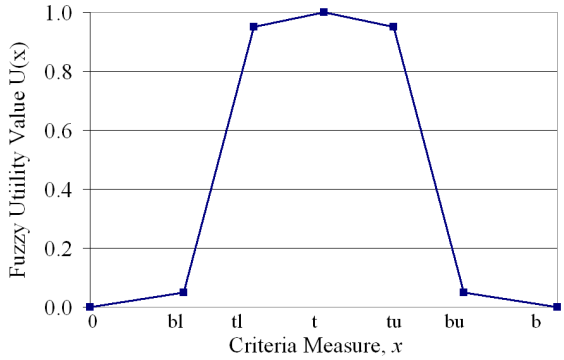


Figure 4.10: Example Piecewise Linear Fuzzy Utility

4.5.1.1 Required Area

The Required Area Satisfaction RAS is the ratio of the space's acquired area AA to its required area RA, eq. 4.4

$$RAS_i = \frac{AA_i}{RA_i} \tag{4.4}$$

A space with more area than necessary, RAS > 1, is more comfortable. However, the extra acquired area is likely the result of another space's shortage of space. If a space is too small, it may no longer be able to provide the intended function. The control points, Table 4.5, and plotted utility, Figure 4.11, for the default required area fuzzy utility are as follows:

Table 4.5: Required Area Fuzzy Utility Default Control Points

RAS_i	0	0.50	0.98	1.10	1.50	2
U_{RAS_i}	0.0	0.05	1.0	1.0	0.05	0

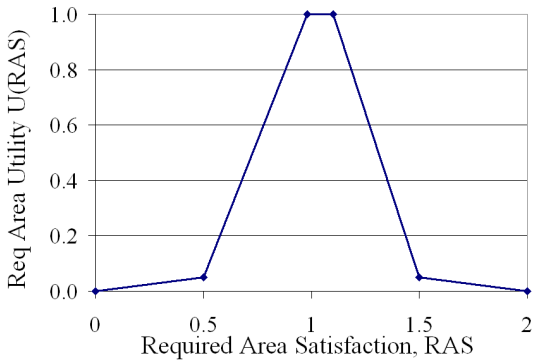


Figure 4.11: Default Required Area Fuzzy Utility

4.5.1.2 Aspect Ratio

Aspect Ratio AR is the ratio of overall dimensions, Δx and Δy, from all j boxes of space i, eq. 4.5. Figure 4.12 illustrates Δx, Δy, and minimum segment dimension MSD. The most useful spaces have an aspect ratio near 1. Large deviations from one result in inefficiently shaped spaces. In the default utility, spaces with an Aspect Ratio between $\frac{1}{3}$ and 3 are considered acceptable. The control points for the default Aspect Ratio fuzzy utility are shown in Table 4.6.

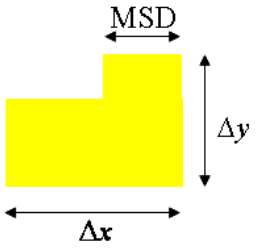


Figure 4.12: Sample Shape with Overall Dimensions and Minimum Segment Dimension MSD

$$AR_i = \frac{\Delta x}{\Delta y} = \frac{\max(Fore_j) - \min(Aft_j)}{\max(Starboard_j) - \min(Port_j)} \tag{4.5}$$

Table 4.6: Aspect Ratio Fuzzy Utility Default Control Points

AR_i	0	0.30	0.33	3.00	3.30	6.0
U_{AR_i}	0.0	0.05	1.0	1.0	0.05	0

4.5.1.3 Minimum Overall Dimension

The lesser of Δx and Δy is taken as the minimum overall dimension MOD, eq. 4.6.

$$MOD_i = \min(\Delta x, \Delta y) \tag{4.6}$$

This criterion is used to ensure adequate room for any required large machinery or equipment. The minimum overall dimension fuzzy utility is also a reflection of aspect ratio because utility is calculated in terms of required area. The default control points are as follows in Table 4.7. The resulting default minimum overall dimension fuzzy utility is shown in Figure 4.13.

Table 4.7: Minimum Overall Dimension Fuzzy Utility Default Control Points

MOD_i	0	$0.5\sqrt{RA_i}$	$0.55\sqrt{RA_i}$	$> 0.55\sqrt{RA_i}$
U_{MOD_i}	0.0	0.05	1.0	1.0

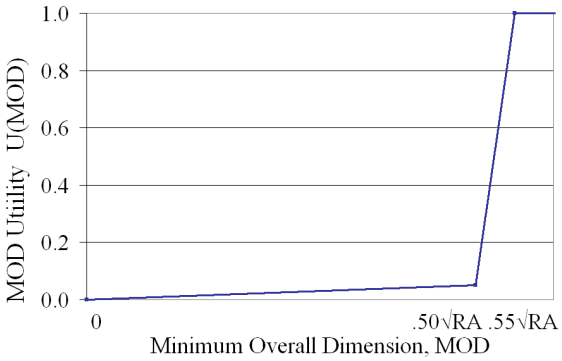


Figure 4.13: Default Minimum Overall Dimension Fuzzy Utility

4.5.1.4 Minimum Segment Dimension

The flexibility of the three box approach requires additional controls. The minimum segment dimension (MSD, Figure 4.12) of space's shape must be checked. To ensure usability, no extension from the compartment can be narrower than the minimum dimension MD. The default minimum dimension is 2.0 m enabling an entryway, if required for access. Appendages only grow into a FreeSpot wider than or equal to the minimum width, but the smallest width must be at least wider than 1.2× MD to be fully satisfied as defined by the default control points, Table 4.8. The resulting default minimum segment dimension is shown in Figure 4.14.

Table 4.8: Minimum Segment Dimension Fuzzy Utility Default Control Points

MSD_i	0	1	1.2× MD	> 1.2× MD
U_{MSD_i}	0.0	0.05	1.0	1.0

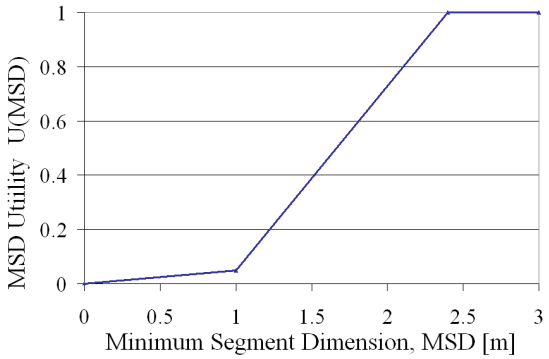


Figure 4.14: Default Minimum Segment Width Fuzzy Utility

4.5.1.5 Perimeter

Some L and T shaped spaces can satisfy all the required area, aspect ratio, and minimum dimension criteria, but they are still not an ideal shape. This was discovered in early experimentation that yielded spaces composed of two nearly square elements with a very small overlap. This imperfection is caught by controlling

the perimeter. The default perimeter fuzzy utility control points are shown in Table 4.9. An additional control point may be added to the polyline at perhaps ($10\times\sqrt{\text{RA}/3}$,0.1) if the user finds the default too lenient.

A square has the shortest perimeter to enclose a given volume using only orthogonal grid lines. Therefore, to foster square shapes, the perimeter fuzzy membership function gives perfect utility to a space with a perimeter approximately equal to $4\times\sqrt{\text{RA}}$. Utility should drop both above and below the ideal length. The default shortest fully satisfied perimeter is 98% of the perimeter of a square of the required area. The longest fully satisfied perimeter control point is for a space with perfect area satisfaction and a 3:1 aspect ratio. A longer perimeter results from shapes with poorer aspect ratio or from the appendages. Where RA_l is a length with the same value of the required area RA, a space with dimensions RA_l by 1 has the longest possible perimeter $2RA_l + 2$ for the RA. Shorter perimeters are found from spaces that do not meet their required area constraints. However, a small and awkwardly shaped space can still satisfy the perimeter constraint. Therefore, the combination of all the constraints is needed to fully capture what mathematically constitutes a functional shape.

Table 4.9: Perimeter Fuzzy Utility Default Control Points

Per_i	1	$0.98\times4\sqrt{RA_i}$	$8\sqrt{RA_i/3}$	$2RA_l + 2$
U_{Per_i}	0.0	1.0	1.0	0.0

4.5.1.6 Adjacency and Separation

Spaces can have adjacency and/or separation constraints relative to other spaces or locations. Proximity constraints are evaluated for each space in the current Zonedeck. For two related spaces, the separation in the x (longitudinal) and y (transverse) directions are found using the two closest points between them. This gives a conser-

vative approximation for separation. If the target space is in a Zone-deck that has not yet been arranged, the closest point of the Zone-deck to which it has been allocated is used for the target's location. However, since the allocation has already assessed separation and adjacency in the increments of Zone-decks, only proximity constraints between spaces in same Zone-deck or in adjacent Zone-decks are calculated. This smaller distance scale is for fine tuning.

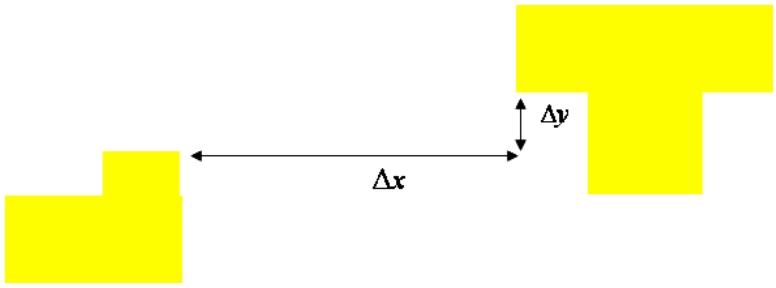


Figure 4.15: Sample Proximity Distance by Closest Points

The distance, d, between the spaces is calculated by either a Euclidean or Manhattan formulation using Δx and Δy as shown in Figure 4.15. Euclidean (eq. 4.7) is the straight line length between the two closest points, for example for a noise buffer from machinery to accommodations. If the travel path is of interest, as in wave guide length limitations, the Manhattan formulation (eq. 4.8) can be used. The default control points for the piecewise linear fuzzy utilities for adjacency and separation constraints are shown in Table 4.10.

$$d_{Euc} = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{4.7}$$

$$d_{Man} = \Delta x + \Delta y \tag{4.8}$$

4.5.1.7 Access Connectivity Separation

To qualify as an accessway, the connectivity to the passage must be wide enough for a door (approximately 1m). The first connectivity for every space is set early

Table 4.10: Adjacency and Separation Fuzzy Utility Function Default Control Points

d	0.0	0.50× Beam	2× Z-d Length	Ship Length
U_{Adj_i}	1.0	0.95	0.05	0.0

d	0.0	1	0.50× Beam	Ship Length
U_{Sep^i}	0.0	0.5	0.95	1.0

in the arrangement and preserved. On the Below DCD Zone-decks, the second connectivity is also mandated from the beginning. However, spaces that require more than one access connection on the DCD must be evaluated to make sure the second connection is present and a sufficient distance, d, apart from the first one. Figure 4.16 shows how distance is calculated for three cases (left to right): a space connected to a longitudinal and an athwartships passage, a space connected onto two longitudinal passages, and a space with access on only one longitudinal passage. Appendages also contribute to the separation distance. The default access connectivity separation fuzzy utility control points for access separation distance are shown in Table 4.11.

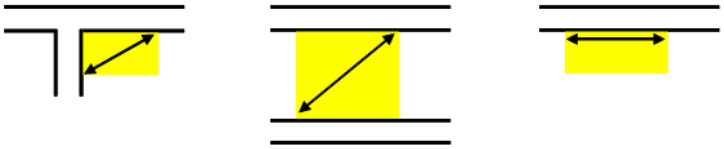


Figure 4.16: Separation Distance Between Two Accesses on DCD

Table 4.11: Access Separation Fuzzy Utility Default Control Points

d	0	2× MD	$\sqrt{RA_i}$	$\sqrt{RA_i}$	$\geq \sqrt{RA_i}$
U_{Acc_i}	0	0.05	0.95	1.0	1.0

4.5.2 Definition of the Objective Function

The goal is to find the maximum utility value for a given topology for the current Zone-deck, U^*. Minimum correlation fuzzy inference is used to make the decision

(Kosko 1992). The objective function defined in eq. 4.9 calculates the overall utility for the geometry from the fuzzy utility values. For each space i, the minimum utility value from all criteria is taken into an average over all the N spaces.

$$U = \frac{\sum_i^N \ \min(U_{RAS_i}, U_{AR_i}, U_{MOD_i}, U_{MSD_i}, U_{Per_i}, \text{ all } U_{Adj_i}, \text{ all } U_{Sep_i}, U_{Acc_i})}{N} \quad (4.9)$$

Each run of the geometry generation routine results in a different solution due to the randomness applied to the space, growth, and direction selection. After multiple runs, the candidate topology's best geometry utility U^* is established by exhaustive search and returned as the topology optimization cost function value.

4.6 Optimization Method

A Genetic Algorithm is applied to each Zone-deck to establish the topology that yields the highest utility geometry. Figure 4.17(a) again shows the double loop structure that alternates between the topology chromosome manipulation and the geometry cost function in each generation of the GA. The corresponding expanded steps of the Part II GA are included in Figure 4.17(b).

4.6.1 Variation Operations

Arrangement GA variation operations are tailored to preserve the allocation solution to the DCD Sub-Zone-decks and to the Below DCD Zone-deck. On the Damage Control Deck, crossover and swap are both applied. Crossover is adapted to only allow crossover points between chromosome Sub-Zone-deck segments. One segment and two chromosomes are selected randomly. Then the two entire segments are switched between the two chromosomes. Technically, this becomes a two-point crossover if the second (middle) gene segment is exchanged. Using this adaptation, the contents of each Sub-Zone-deck in all chromosomes as established in the allocation are maintained. As there are no Sub-Zone-decks segments below the DCD, the

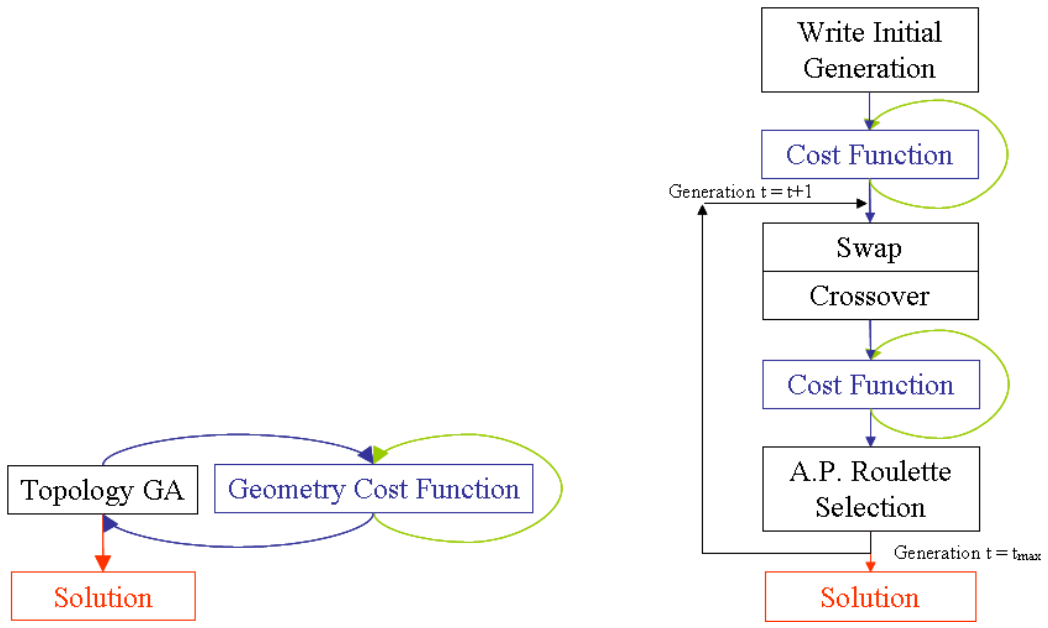


Figure 4.17: Arrangements Optimization Expanded Schematic

crossover operation is not used for below DCD Zone-decks since this would invalidate the allocation.

Swap is applied for both DCD and non DCD Zone-decks, but on the DCD Zone-decks, swap is also altered to only manipulate spaces within one Sub-Zone-deck segment at a time. When swap is called, one of the Sub-Zone-decks (SZDs) is picked by a random draw translated to selection probabilities. The port and starboard SZDs each have a 20% chance of selection, and the center SZD is given a probability of 60% as it typically will have more spaces to re-arrange. For below DCD chromosomes, no chromosome segment selection is necessary; the entire chromosome is considered one SZD segment. The two swapped spaces are randomly selected from within the selected gene segment. There is, however, a catch to make sure that the athwartship passage in the center Sub-Zone-deck is not swapped into the first or last position. For better algorithm performance, at least one space resides in the bow and in the stern section. A random draw between one and the selected segment's length

determines how many exchanges are performed within the segment each time swap is called. Since fewer variation operators are used compared to the allocation GA, these additional exchanges help produce the desired amount of diversity in the population.

Due to the high computational cost of the geometry generation, the population number (number of chromosomes in the population, PopNum) is small and daughter chromosomes do not replace the parents. Instead, daughter chromosomes are placed in a second, temporary population. The number of daughters collected in the temporary population is equal to the population number. Each crossover operation produces two daughters, and swap produces one daughter. Therefore, to give equal representation to both operations, GAs optimizing a DCD Zone-deck perform crossover PopNum/4 times and swap is called PopNum/2 times each generation. When acting on a non DCD Zone-deck, the number of swap calls is equal to the PopNum. The population number should be a multiple of 4 so that the number of operations calculated is an integer. These statistics are included in Table 4.12.

Table 4.12: Number of Swaps and Crossovers per Generation by Zone-deck type

	DCD	non DCD
Crossover	PopNum/4	0
Swap	PopNum/2	PopNum

4.6.2 Adaptive Probability Roulette Selection

The selection operation selects chromosomes from both the original population and the temporary population. Selection probabilities for all 2×NumPop chromosomes are again calculated by summing and normalizing the utility values to find the fitness percentages ($Psel$) and the corresponding Roulette probability values ($Rsel$). The elite chromosome is automatically included in the next generation. The small population is more susceptible to losing diversity by repeated selection of the fitter

chromosomes. Adaptive Probability Roulette was, therefore, written to ensure that a single chromosome can only be selected once to survive to the next generation.

After a chromosome is selected, its probability of selection percentage ($Psel$) is subtracted from the total and then re-written to zero. The $Rsel$ values of the chromosomes following in the list are re-calculated, and the next random draw is taken from between zero and the new decreased total. This repeats until NumPop chromosomes have been selected for the next generation's population.

4.7 Example Problem

4.7.1 Inputs

The topology/geometry arrangements optimization algorithm is illustrated by a proof-of-concept example for one subdivision with a Damage Control Deck (DCD) Zone-deck and an associated below DCD Zone-deck. Contents of each Sub-Zone-deck as an assumed solution from the Allocation problem are shown in Table 4.13 with the spaces' area and number of accessway requirements. The total desired area of the spaces allocated to each Sub-Zone-deck reasonably matches the available area of the Sub-Zone-deck through the allocation process.

Both Zone-deck footprints are 24m by 24m with a 1m grid spacing. The port stairtower (shown later in grey) is set outside the port longitudinal passage in the port DCD Sub-Zone-deck, while the starboard stairtower is assigned to be on the inside of the starboard passage. Compliant with Naval Sea Systems Command (1992), passages are 1m wide, and the stairtowers are longitudinally oriented, 3m long by 1m wide. The available area given in Table 4.13 reflects the subtraction of these dimensions.

An additional pre-defined fixed object (perhaps representative of a trunk) is accounted for in the port SZD geometry (shown later in black) to show that SZDs not

Table 4.13: Allocation Solution applied to Arrangement example problem

Space Number	ReqArea [m2]	AccNum
DCD, Subdivision 1		
Port SZD Available Area: 141 m^2		
1	54	1
2	48	1
3	42	1
Port SZD Assigned Area: 144 m^2		
Center SZD Available Area: 227 m^2		
4	20	1
5	48	1
6	15	1
7	30	1
8	80	2
9	36	1
Center SZD Assigned Area: 229 m^2		
Stbd SZD Available Area: 144 m^2		
10	36	1
11	36	1
12	36	1
13	36	1
Stbd SZD Assigned Area: 144 m^2		
DCD, Subdivision 0		
14	n/a	n/a
DCD, Subdivision 2		
15	n/a	n/a
Below DCD, Subdivision 1		
Below SZD Available Area: 564 m^2		
16	134	1
17	80	1
18	56	1
19	60	1
20	60	1
21	115	2
22	60	1
Below SZD Assigned Area: 565 m^2		

arranged by the Stochastic Growth Loop can also handle blockages. The area of the blockage is, however, not accounted for in subdividing the outer SZD, because trunk size is assumed to be small compared to the spaces. Therefore, the space(s) around the blockage are may be short on area satisfaction. A further refinement to account for this area loss is appropriate. While it is not possible to handle every intricacy of arrangement choices, the most common complexities are included.

Adjacency and separation constraints are enumerated in a connectivity matrix, Table 4.14. To more strongly express constraints, they are expressed reciprocally. Hence the matrix is symmetric across the diagonal. A proximity constraint to self, marked with an X, is not meaningful. A value 1 indicates a preference for adjacency; 2 indicates separation. Two additional spaces are included to show the ability to relate to objects outside of the current Zone-deck. Space 14 was allocated to the next Zone-deck forward, and assumed to already be arranged. Space 15 is assumed to reside in the next, un-arranged Zone-deck aft.

Table 4.14: Connectivity Matrix for DCD

	Port SZD			Center SZD						Stbd SZD				Fwd	Aft
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	X	1		1				2			1			1	
2	1	X					1								
3			X												1
4	1			X	1	1			2					1	2
5				1	X	1									
6				1	1	X			1						
7		1					X					1	1		
8	2							X						2	
9				2		1			X	1			1		
10									1	X					
11	1										X	1			
12							1				1	X			
13							1		1				X		
14	1			1	2									X	
15			1	2											X

There are 40 adjacency and separation constraints, $5\times(3 + 6 + 4) = 65$ shape

constraints, and one extra double access constraint for a total of 106 fuzzy utilities to calculate in the DCD cost function. With six spaces and an arrangeable athwartships passage in the center Sub-Zone-deck there are 7! = 5040 possibilities. However, since the athwartships passage will not be in the first or last position, the center SZD search space is reduced by 2×6!=2×720=1400 to 3600 giving a total search space of 3!×3600×4! = 518,400 variations of the independent variable. The below DCD has 7! = 5040 possible topologies.

4.7.2 Results

4.7.2.1 Damage Control Deck

The GA improved the arrangement solution by 28% from the first generation. Convergence is seen in Figure 4.18. The initial arrangement is seen in Figure 4.19(a) and has a utility value of 0.7093, eq. 4.9. Below it is the final solution, Figure 4.19(c). The best fitness of 0.9112 was found at generation 87 of 100. This example written in object oriented C++ ran in 192 seconds (3 minutes and 12 seconds) with a population of eight chromosomes and five geometries generated per topology on a 1.73 GHz PC with 1 GB RAM.

A noticeable improvement can be seen in the final solution. Most obviously, there are no longer any open spaces. Exiting at the maximum iteration number with grid spots left unoccupied does not directly affect the cost function value, but often, as in this case, some spaces have less than their required area and the user may decide to manually repair the solution. As reported in Table 4.15, there is a good match between acquired area and required area in the final solution.

Strong, practical shapes were generated in both solutions. The only irregularity is final Space 9's small extension to fill in the area next to the stairtower. Though appendages are not desirable, they are necessary to complete more complex design

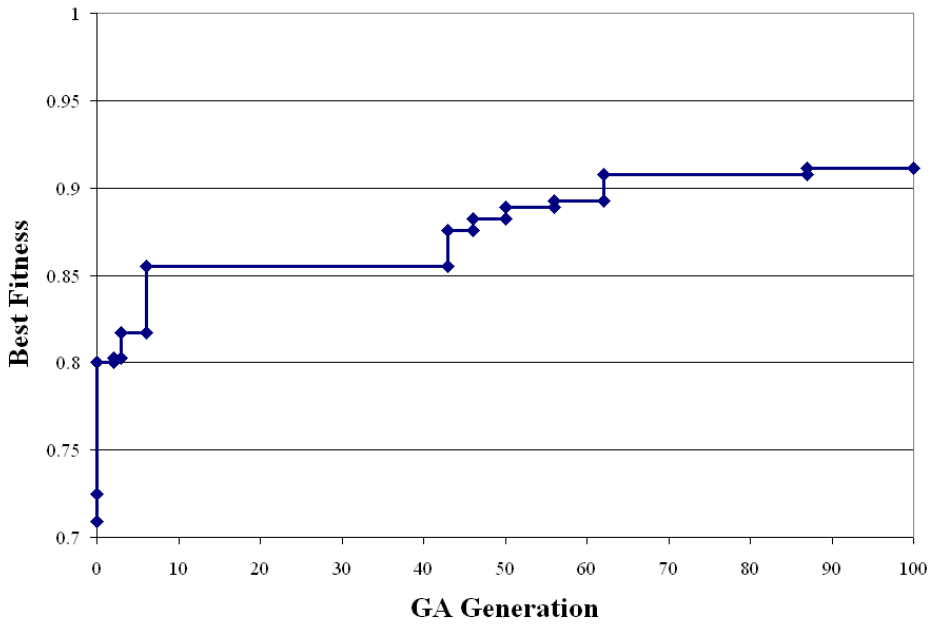


Figure 4.18: Convergence Plot for DCD GA

Table 4.15: Area Satisfaction from First and Final DCD Arrangement

Space	First	Final	Required Area
1	51	47	54
2	44	48	48
3	42	42	42
4	15	20	20
5	49	50	48
6	15	16	15
7	22	30	30
8	80	74	80
9	25	37	36
10	36	36	36
11	36	36	36
12	36	36	36
13	36	36	36

problems that consider vertical and horizontal passages, stairtowers, and fixed objects.

Space 8 was required to have two passage connectivities. It is fully satisfied by the first solution since the space lies between both longitudinal passages. In the final solution, the separation distance between accessways can be greater with attachments

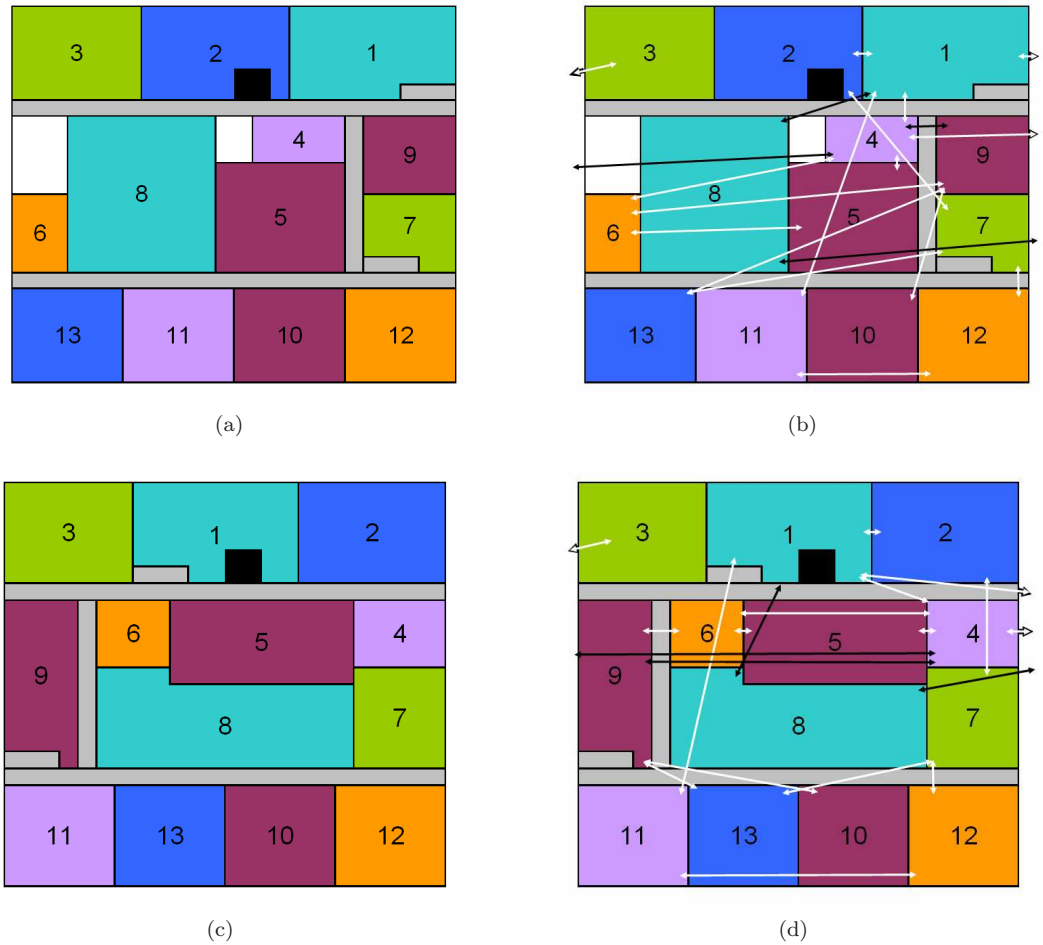


Figure 4.19: DCD First (a and b) and Final (c and d) Solutions showing Adjacencies (white) and Separations (black)

to the starboard and athwartship passages. The resulting thinner aspect ratio (7:3) is not as good, but it is still within the acceptable 3:1 range. Also, by not attaching to the port passage, space 8 improved its utility for the separation constraint from space 1 in the port SZD.

The final solution greatly improved the adjacency and separation goals, Table 4.14. Figures 4.19(b) and 4.19(d) overlay arrows depicting these goals on the solutions. For example, Space 6's three adjacency (shown in white) lengths, which were prominent in the first solution, are shortened in the final. Placing 9 in the stern section and 4 towards the bow helped satisfy the six proximity constraints on 4 including separations (in black) to 9 and to 15 and an adjacency to space 14. Arrows extending out of the Zone-deck to the right are representative of a relationship to space 14, and similarly arrows to the left are to space 15 in the aft Zone-deck. The topology optimization is clearly responsive to proximity criteria.

The stairtower configuration established in the final DCD solution is brought down to the below DCD Zone-deck optimization as a fixed constraint.

4.7.2.2 Below Damage Control Deck

The Below Damage Control Deck case proved to be a more difficult problem. Not only is the available area handled at any one time much larger, but it also has variable blockages within the area from the fixed stairtowers and associated passages. The example Zone-deck is also over-allocated. The example problem is written to demonstrate how well the algorithm handles non-ideal and realistic scenarios. At least $2(2\times3)=12$ of its 576m^2 are dedicated to accessways leaving only 564m^2 for the total 565m^2 of required area assigned to it. Therefore, it's important to keep passages short and to fill up all of the available area. Frequently, however, the stochastic growth loop exited at the maximum number of iterations rather than

converging to no area remaining unoccupied. Therefore, the sensitivity to maximum iterations was a key concern.

To find reasonable solutions, the maximum geometry generation iterations and the number of geometries generated per topology had to be increased from the DCD problem. A study was conducted to establish how these two parameters affect the resultant cost function in order to find the optimal geometry generating parameter values. The highly stochastic nature of the algorithm mandates repetition to cull out poor solutions, but the algorithm also reaches a point of diminishing returns the longer the optimization runs. The goal is to find the minimum parameters necessary to find strong solutions before computation time becomes excessive.

Two topology cases were considered. Running the optimization with all spaces requiring a single access gave the first test chromosome. The second chromosome was the GA's elite from a case with two spaces required to have double access. For each case, ten geometries were made for six different maximum iteration caps ranging from 2500 to 20000. The average and maximum of the ten fitness values were plotted versus maximum iteration, Figure 4.20. The right hand axis, Average

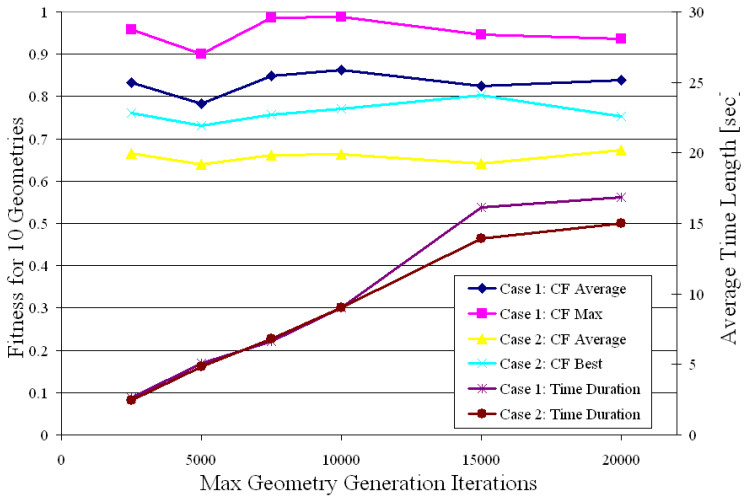


Figure 4.20: Cost Function and Elapsed Time versus Maximum Geometry Generation Iteration for Case 1 and 2

Time Length [sec], corresponds to the bottom two lines labeled Time Duration. As expected, duration monotonically increases with maximum iteration. However, the cost function values do not. Fitness is remarkably independent of maximum iteration. This justifies the use of the smallest maximum iteration that still generates strong solutions.

The smallest maximum was determined by the parameter's relationship with remaining grid spots. There is only a loose, inverse relationship between cost function and remaining spots, Figure 4.21. Yet, it is still desirable to produce a solution that requires less repair work by the designer to allocate the unoccupied remaining spots to spaces.

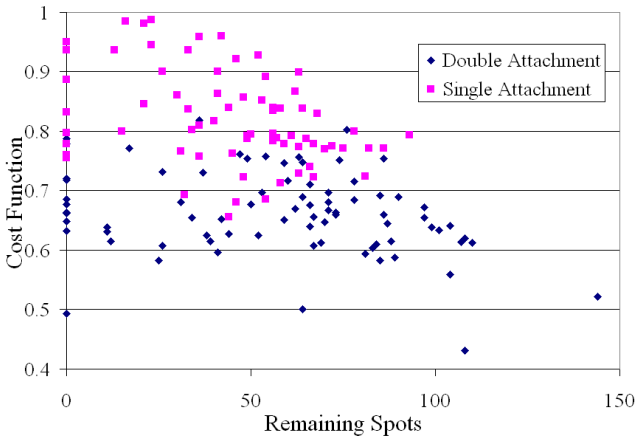


Figure 4.21: BDCD Parametric Study: CF vs Remaining Spots

The case for choosing a maximum of 7500 iterations is supported by a plot of remaining spots versus maximum iteration number, Figure 4.22. There was also an increase in average cost function at 7500 iterations, Figure 4.20. Therefore, to balance computational time and performance, a maximum iteration limit of 7500 was placed on the below DCD Zone-deck geometries for the example problem.

A second parametric study looked at how many geometries should be generated each time the GA calls the cost function. Over 15 runs of ten geometries each,

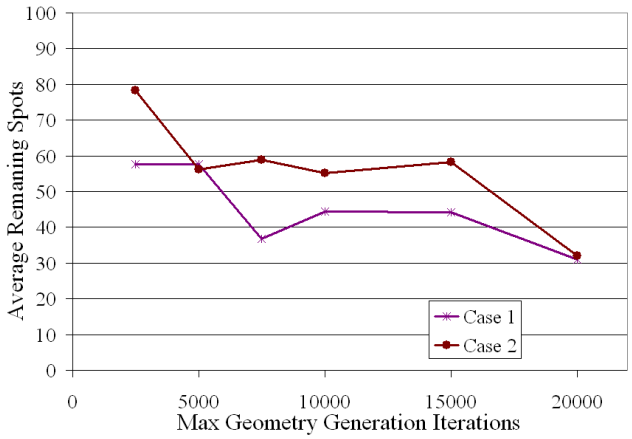


Figure 4.22: BDCD Parametric Study: Remaining Spots vs Iteration

the best geometry appeared, on average, at number 6.4. This makes the case for generating all ten geometries each time. If duration becomes a greater concern, the number of geometries could be reduced from 10 to 7.

The following results were produced by a population of 8 chromosomes each generating 10 geometries with a maximum iteration limit at 7500 for each cost function call. Same as on the DCD, 8 chromosomes were chosen to balance population diversity and computation time. Two solutions are given representing the variety produced from different optimization runs on this multi-modal and discontinuous topology search space. Computation times vary from run to run, but roughly each GA generation making 80 geometries takes just over one minute.

A 25 generation run took 1550 seconds (25 minutes, 50 seconds) and found the topology (17 21 20 19 18 16 22) with the top fitness value of 0.8956, Figure 4.23. The corresponding geometry shows perfect satisfaction of the shape criteria and the double access requirement on space 21. However, there is a significant fault: RAS_{16} = 0.75. Space 16 is short $33m^2$, and the remaining spots are not adjacent to it. While this solution gave the best cost function value, it is not easily repaired. It is left to the designer to decide which of the numerically top solutions is best and how

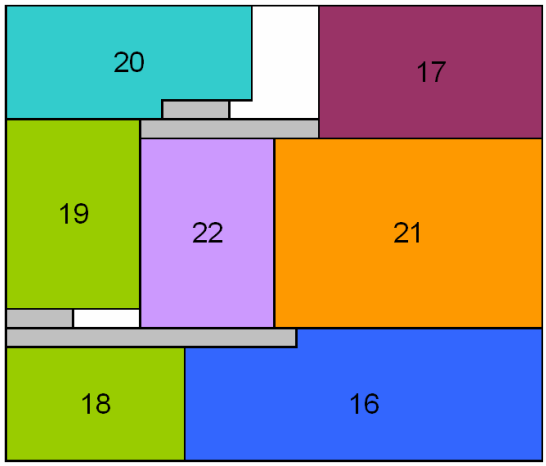


Figure 4.23: Below DCD Solution 1, CF = 0.8956

to repair it if needed.

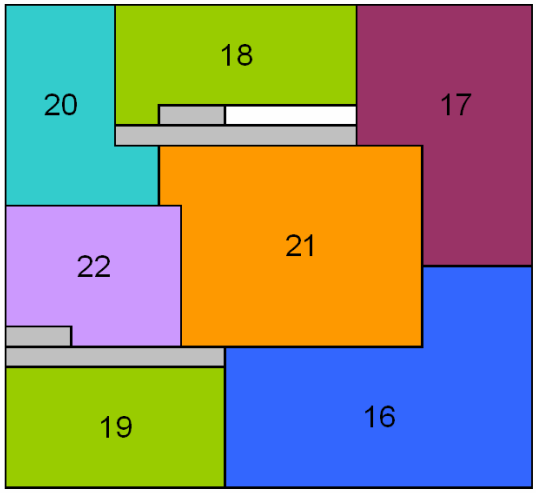


Figure 4.24: Below DCD Solution 2, CF = 0.8855

The best topology (18 17 21 20 16 19 22) as determined by the author has only a 1.1% lower cost function value at 0.8855. The corresponding geometry is shown in Figure 4.24; its two main faults are easily fixed. First, there are open spots between space 18 and the port passage. Space 18's center box is responsible for a side attachment to the passage aft of the stairtower. Therefore, the portion of the space extending forward is an appendage. Since an appendage cannot grow an

appendage in the present algorithm, there is no mechanism for space 18 to grow back toward starboard into the empty grid spots. These spots must be manually allocated for space 18. To keep its almost exact area satisfaction, space 18's forward bulkhead can then shrink by one. This enables space 17 to grow aft and push the passage 1m shorter. The second fault to address is an excess of area in Space 17 and too little area in Space 16. The bulkhead between them is simply translated 2m towards port. Area requirements for the three solutions is given in Table 4.16. After the three repair steps, the resulting geometry, Figure 4.25 has an improved cost function of 0.9179.

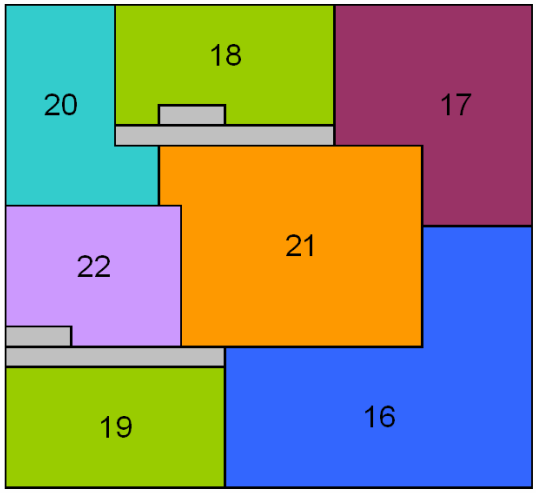


Figure 4.25: Below DCD Repaired Solution 2, CF = 0.9179

Table 4.16: Area Satisfaction from Below DCD Arrangements

Space	Soln 1	Soln 2	Rep Soln 2	Required Area
16	101	118	130	134
17	70	86	81	80
18	54	57	57	56
19	60	60	60	60
20	62	56	56	60
21	120	113	113	115
22	60	53	53	60
Sum:	527	543	550	565

Final assessment is left to the designer for two reasons. First, it is impractical to algorithmically repair the plethora of scenarios that may be produced. Second, it is not the author's intent to remove the designer from the arrangement process. On the contrary, it is essential to include the user's discerning judgment. Visual inspection and real-time expert input are powerful tools that are to be included in the optimization of arrangements.

4.8 Validation Problem

For a full-scale Zone-deck, it is unlikely to attain a perfectly satisfied solution. Compromises between conflicting goals drive the best possible utility lower than one. Yet, to give confidence that this compromised solution is still the best, the algorithm must demonstrate its ability to find a perfectly satisfied solution when there is one.

Three validation problems are presented with "perfect" ($U = 1$) solutions found by alternative optimization methods or other means. The efficacy of the topology optimization by the GA has already been verified in the example problem. Thus, only the geometry generation is validated here. The topology of the known solution is input to the geometry generation routine. For each problem, the routine is run ten times generating up to ten geometries and exiting upon finding a $U = 1$ geometry. Results are given for the average of the ten runs. Twenty-eight of thirty runs were able to generate an optimal solution.

The validation Zone-deck is treated like a center SZD section of the DCD. Spaces require connectivity to either the port or starboard side. Center boxes are placed longitudinally and then expanded to a side access and to the extents. Only shape criteria are evaluated in the cost function, eq 4.9. Proximity and double access constraints were successfully represented in the example problem. The maximum

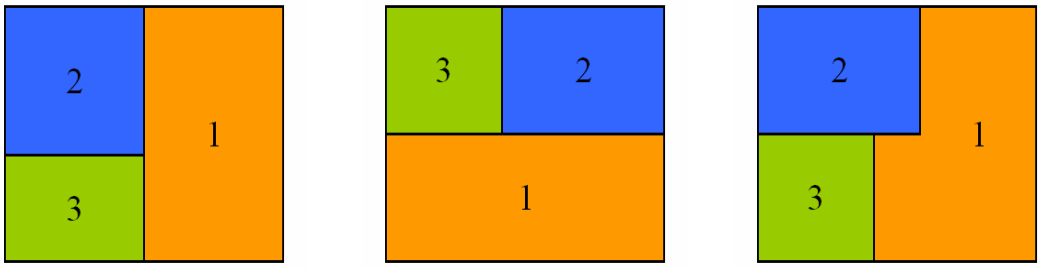
iteration is 2500. The available area is 12m by 12m and each problem has a goal of 100% area utilization.

4.8.1 Three-spaces

The first and simplest problem arranged three spaces. Their required areas are given in Table 4.17. For the given topology, 1-2-3, there is more than one way to arrange a perfectly satisfied geometry. Any $U = 1$ solution was accepted to exit the run. Three of these solutions are shown in Figure 4.26. Variations 4.26(a) and 4.26(b) are more desirable, but the L shape is allowed using the same fuzzy membership functions as applied in the example problem. Over ten runs, a perfect geometry was found, on average, at number 3.5 of ten possible geometries in an average time of 0.192 seconds.

Table 4.17: Three-space Validation Problem Inputs

Space	Required Area
1	72
2	42
3	30



(a) (b) (c)

Figure 4.26: Three-space $U = 1$ Validation Solutions

This problem is also solved by an exhaustive search in a Matlab program. Upper and lower bounds on area and aspect ratio are written for the three spaces. Bounds are defined by the region of the fuzzy utilities equal to one. The 12 inequality

constraints are in terms of two independent variables, x and y. Assuming a layout like Figure 4.26(a), x is the longitudinal length of space 3, and y is the transverse width of space 3. The remaining feasible domain gave a single integer solution of $x = 6$ and $y = 5$. This is the same solution found by the stochastic growth loop. Thus, for this layout with space 1 extending the full beam and space two and three split with a longitudinal joiner bulkhead, the algorithm found the single global optimum.

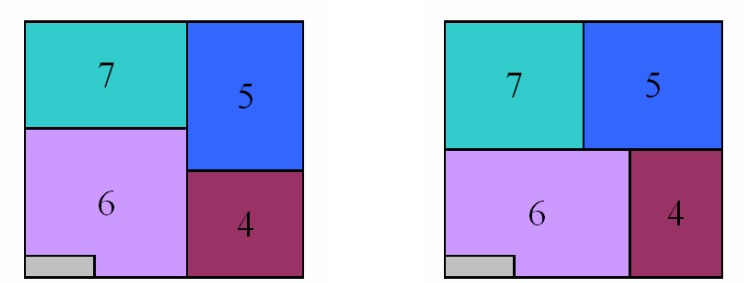
4.8.2 Four-spaces

The 4-space validation problem adds in a fixed blockage, thus, requiring the use of appendages to fill the available area. Problem inputs are given in Table 4.18. The topology is 4-5-6-7. With a maximum of ten geometries each run, not every run found the optimal $U = 1$ solution, Figure 4.27(a). Twice, the routine ended with an alternate strong solution, $U = 0.9892$, Figure 4.27(b). Each space in this solution missed the required area by $\pm$ 1m^2. These two runs made all 10 geometries in search of a $U = 1$ solution. The average number of geometries made was 5.0, and the average duration was 0.055 seconds.

Table 4.18: Four-space Validation Problem Inputs

Space	Required Area
4	25
5	35
6	46
7	35

Introducing one additional space leads to an explosion in the number of variables that could be defined for varied layouts. Even before allowing multiple boxes to accommodate a blockage, there are at least a dozen possible cases. A full enumeration and optimization of all the ways to subdivide the available area might likely reveal more alternate $U = 1$ solutions. However, since another solution cannot improve



(a) (b)

Figure 4.27: Four-space Validation $U = 1$ Solution (a) and Alternate Solution, $U = 0.9892$ (b)

upon a solution with $U = 1$ as defined by this cost function, it is reasonable to find one perfect solution to validate the geometry generation.

A goal of this research has been to create an explicit and objective cost function to define a good arrangement. A mathematical proof or exhaustive search to find a single global optimum would validate the cost function's ability to quantify superior characteristics more than it would serve to validate the ability of the geometry generation to create these characteristics. For the purpose of a preliminary design tool, the first priority is to find a selection of good geometry solutions.

4.8.3 Five-spaces

Two blockages with the dimensions of stairtowers and five spaces were included in the last and most difficult validation problem. The topology is 8-9-10-11-12. From these inputs, Table 4.19, the known $U = 1$ solution (Figure 4.28) was found quickly.

Table 4.19: Five-space Validation Problem Inputs

Space	Required Area
8	15
9	39
10	27
11	42
12	15

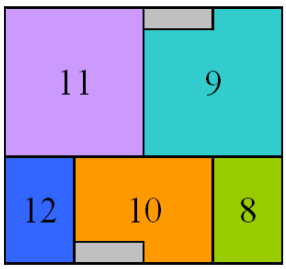


Figure 4.28: Five-space $U = 1$ Validation Solution

4.8.4 Geometry Validated

A summary of all three validation problem's computation statistics is given below, Table 4.20. When a perfect utility is obtainable, the stochastic growth loop can find

Table 4.20: Validation Problem Statistics

Number of Spaces	Average Geoms	Average Time [sec]	Average Time/Geom [sec]
3	3.5	0.192	0.055
4	6.0	0.274	0.046
5	2.8	0.293	0.105

it given multiple attempts. Although the 4-space problem required more geometries, its average time per geometry was less than the 3-space problem. The longest time overall and per geometry was for the 5-space problem, but its number of geometries required was also the lowest at 2.8. The problem with fewer ways to arrange the spaces in a highly satisfied manner required a longer computation time. Whereas, the 3 and 4 space geometries converged quickly, but not always to the absolute best possible geometry. Of 30 runs, the mode of the number of geometries required before successfully producing a perfect utility was two. Thus, although the average is over twice as many, the algorithm typically found the ideal quickly. Six of the 30 runs had to generate ≥ 8 geometries raising the average. The trade-off between performance and speed is balanced within the algorithm. The randomness that at times results in longer computation times is also integral to the robust logic that produces perfect

solutions for problems of varying difficulty. From the strong performance on these three problems, it is reasonable to conclude that for more complex problems, the geometry generation routine is producing the best solutions given the parameters judged to be practical.

CHAPTER 5

Conclusions

5.1 Summary

The algorithms presented take designer inputs for a ship's spaces and return a full-scale general arrangement solution. The problem is broken into two parts: Allocation of spaces to Zone-deck regions in the ship and Arrangement of spaces within each Zone-deck. Spaces are arranged with consideration to global and relative position, required area, aspect ratio, minimum dimensions, and access requirements. Arrangement is performed in two iterative steps. First, topology designates the relative longitudinal and transverse position of the spaces. Second, geometry grows the spaces from these initial seed positions to fill the available area. Fuzzy utilities are used to express the goals and constraints of the design.

For Part I, allocation is solved for an example problem assigning 70 spaces to 17 Zone-decks. The independent variable vector is an ordered list by space index number of each space's assigned Zone-deck index number. The allocation solution evaluates Zone-deck area utilization and spaces' relative and global position goals. Adjacency and separation distance between spaces is measured in increments of deck and subdivision. Global position is assessed by deck and subdivision. Each discrete distance and position has an editable default fuzzy preference value.

A Genetic Algorithm optimizes the integer-coded chromosome. In addition to

simple crossover and mutation, Simulated Binary Crossover (Deb 2001) and a new operation, swap, diversify the population. Swap exchanges the placement of two randomly selected spaces as is particularly required in later generations to maintain proper area utilization. A Double-round Tournament Selection converges the population. In approximately two minutes, the Fortran 90 code performed 4000 generations. The final elite allocation chromosome had excellent Zone-deck area utilization and optimized fitness for spaces' global and relative placements.

For part II, two typical Zone-decks in one subdivision of a sample ship are arranged. The relative longitudinal and transverse position of each space's seed location given by a topology chromosome is translated onto an orthogonal grid. In the geometry step, spaces are expanded to have size and shape filling the available area in the stochastic growth loop. Spaces are defined by up to three contiguous boxes allowing for L, T, C and Z shapes. The Zone-decks are arranged one at a time in sequence.

The Damage Control Deck (DCD) Zone-deck is arranged first. It is divided into three Sub-Zone-decks (SZDs) by the fixed port and starboard longitudinal passages. Vertical stairtower longitudinal positions are optimized by the algorithm and then fixed for the below DCD Zone-deck that is arranged second. There, passages have variable length controlled by the spaces growing and shrinking. Spaces maintain connectivity to one or two passages as required. There are no SZDs in the below DCD Zone-deck; the entire available area is arranged at once.

Converged geometries are evaluated in the arrangement cost function. This evaluates each space's required area satisfaction, aspect ratio, minimum overall dimension, minimum segment dimension, perimeter length, connectivity to access, and again, proximity constraints to other spaces. Editable piecewise linear fuzzy utility functions translate each criteria measure to a fuzzy utility. For each Zone-deck, the best

of a modest number of geometry solutions returns all the bulkhead locations and a cost function value to a Genetic Algorithm optimization of the topology chromosome.

The objective of the topology GA is to find the best corresponding geometry. Tailored swap and crossover variation operators are applied to preserve the allocation solution to the DCD Sub-Zone-decks and to the Below DCD Zone-deck. Adaptive Probability Roulette Selection ensures that a single chromosome can only be selected once to survive to the next generation. The geometry step provides the cost function for the topology optimization.

The DCD Zone-deck GA improved the example arrangement solution by 28% from the first generation. This example written in object oriented C++ ran in approximately three minutes for 100 generations with a population of eight chromosomes and five geometries generated per topology. The final solution shows a great improvement in adjacency and separation constraints. Both the DCD and non DCD Zone-deck algorithms produce spaces with practical shapes and required area satisfaction. The below DCD GA runs approximately one minute per generation due to the larger arrangeable area and the variable length passages. A final below DCD Zone-deck geometry was manually repaired to allocate left-over area and to improve two spaces' required area satisfaction by shifting their shared bulkhead. The resulting geometry is the best as deemed by the author.

5.2 Intellectual Contributions

The success of this work is a result of novel approaches. Access maintenance and shape flexibility are key advances on previous work. The following list enumerates the new contributions to arrangements optimization.

- The two part allocation and arrangement optimization can handle a full scale ship from a space database and main hull structure and return spaces' bulkhead

locations.

- The integer-coded allocation chromosome automatically assigns each space to uniquely one discrete Zone-deck. From this efficient formulation, alternative optimization methods may be attempted, for example, the hybrid GA-Agent approach.
- Complex shapes needed in realistic designs are possible with the three-box approach.
- Probabilistic move selections attempted with heuristic move rules make the stochastic growth loop robust and effective in generating converged geometries.
- Spaces may require connectivity to port and/or starboard passage for single or double access. Continuity is assured for longitudinal passages along decks and for vertical stairtowers in subdivisions.
- Fuzzy logic is exemplified for arrangements optimization.
- The optimization cost function provides a quantitative criterion for the assessment of arrangements that can lead to a more objective, repeatable process.
- Constraint definitions capture best practices of the aging workforce and its accumulated experience. The constraint database is transferable for use on similar designs.

The goal to provide the naval architect with a semi-automated tool has been satisfied. The allocation and arrangement optimization replaces significant man hours, allowing the designer to generate and evaluate more design alternatives in less time with readily available computing power. As design is a highly iterative process, the

designer remains in the loop by editing goals and constraints, evaluating the solutions produced, and then by re-editing and arranging again.

5.3 Future Work

There are more variations of types of Zone-decks than demonstrated here in the development and example problems. DCD Zone-decks in the bow and stern subdivisions typically have different passage configurations, and thus different Sub-Zone-decks (SZDs). A combination of the stochastic growth loop and the approach taken for the port and starboard SZDs can be applied, but the detailed implementation has not been included in this work. Smaller available areas with fewer allocated spaces may actually be left to the designer to simply subdivide. Specific treatment of the superstructure is also left to later work, but it may be treated with the same fundamental algorithms as used on the DCD. Zone-decks on the DCD with no athwartship passage are also not yet coded, but no new logic is necessary to accommodate these. It is simply an extension of the work already presented.

More substantial adaptations will have to be written to handle non-orthogonal entities. Hulls are not orthogonal, but curvature drawn on a u-v space can be mapped to the orthogonal x-y grid. Then, every grid spot is not $1m^2$; a more explicit available area calculation will need to be made. Slanted passages can be also be accounted for by re-defining the available area of the grid spots that are truncated. The stochastic growth loop presented can then be applied on the grid overlaying the available area.

Some modular spaces with fixed dimensions (such as used in some staterooms) should be treated differently from the current stochastic growth loop rules. The user could re-write the fuzzy utility shape criteria functions to only allow a tight margin of error around the pre-defined shape. However, this approach would be expected to result in many poor solutions. Instead, these spaces can be given proper shape

in the center box expansion stage before the growth loop. A positive growth in one direction would then have to be paired with a negative growth in the opposite direction to achieve translation.

A future paradigm shift in the independent variable formulation could better allow for other types of movement. It could be advantageous to translate spaces and to enable one space to push or pull another. Variables could define the bulkhead location between two spaces rather than, or in addition to, defining the bulkhead for one space. This challenge is left to longer term future work.

APPENDICIES

APPENDIX A

Abridged Best Chromosome History with Creating Operations

GA search history of the best allocation chromosome is shown along with the generation of introduction and associated fitness value, Figure A.1. The genetic operations that resulted in the new best chromosome are shown at the right. Where the genetic changes are visible in the abridged chromosome, the changes are highlighted by boxes. Note that more than one genetic operation operated on most of the new best chromosomes. The search terminated at generation 3717 following five reseedings with no further progress. The first allele in the first best chromosome indicates that space 1 is assigned to Zone-deck 13. The box between generation 5 and generation 6 indicates that for the new best chromosome created in generation 6 one of the three mutations or the two space swap moved space 3 from Zone-deck 5 to Zone-deck 9.

Generation	Fitness	1	2	3	4	5	6	7	8	9	...	61	62	63	64	65	66	67	68	69	70	
1	0.0218	13	10	6	16	1	17	16	2	2		9	11	8	14	7	4	4	6	15	1	New
3	0.0375	6	15	5	5	16	2	14	2	7		10	16	4	5	1	6	7	12	1	1	New
4	0.0557	6	15	5	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	Crossover
5	0.0560	6	15	5	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	1 Mutate
6	0.1138	6	15	9	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	3 Mutate, 1 Swap
7	0.1977	6	15	8	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	2 Mutate
8	0.1982	6	15	9	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	2 Mutate, 2 Swap
9	0.1987	6	15	9	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	4 Mutate, 1 Swap
10	0.1993	6	15	9	5	1	17	16	2	2		10	16	4	5	1	6	7	12	1	1	3 Mutate, 1 Swap
12	0.2024	6	15	9	5	1	17	16	2	2		2	16	4	5	1	6	7	12	1	1	1 Swap
15	0.2595	6	15	9	5	1	17	16	2	2		2	16	4	5	1	6	7	12	1	1	1 Mutate
16	0.2601	6	15	9	5	1	17	16	2	2		2	16	4	5	1	6	7	12	1	1	1 Mutate, 1 Swap
18	0.2608	6	15	9	5	1	17	16	2	2		2	16	4	5	1	6	7	12	1	1	4 Mutate
19	0.2662	6	15	9	5	1	3	16	9	2		2	16	4	5	1	6	7	12	1	1	8 Mutate
20	0.2894	6	15	9	5	1	17	16	11	2		2	16	4	5	1	6	7	12	1	1	9 Mutate, 1 Swap
23	0.2908	6	15	9	5	1	17	16	11	2		2	16	4	3	1	6	7	12	1	1	2 Mutate, 1 Swap
24	0.2945	6	15	9	5	1	17	16	11	2		2	3	4	5	1	6	7	12	1	1	2 Mutate
26	0.3085	6	15	9	5	1	17	16	11	2		2	3	4	5	1	6	7	12	3	1	2 Swap
28	0.3124	6	15	9	5	1	17	16	11	2		2	3	4	5	1	6	7	12	1	1	3 Mutate, 2 Swap
29	0.3196	6	15	9	5	1	17	16	11	2		2	3	4	5	1	6	7	12	3	1	4 Mutate, 2 Swap
32	0.3318	6	15	9	5	16	17	16	11	2		2	3	4	5	1	6	7	12	3	1	2 Mutate
33	0.3335	6	15	9	5	16	17	16	11	2		2	3	4	5	1	6	7	12	3	1	2 Mutate, 1 Swap
34	0.3675	6	15	9	5	16	17	16	11	2		2	3	4	5	1	6	7	12	3	1	5 Mutate
35	0.3721	6	15	9	5	16	17	16	11	2		2	3	4	5	1	6	10	12	3	1	1 Mutate, 1 Swap
36	0.3929	6	15	9	5	16	17	16	17	2		2	3	4	5	1	6	10	12	3	1	1 Swap
39	0.3973	6	15	9	5	16	17	16	17	2		2	3	4	5	1	6	10	12	3	1	1 Mutate
41	0.3992	6	15	14	5	16	17	16	17	2		2	3	4	5	1	6	10	12	3	1	1 Mutate
42	0.4173	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	10	12	3	1	5 Mutate
43	0.4232	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	1 Swap
45	0.4318	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	4 Mutate
47	0.4328	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	1 Swap
48	0.4338	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	1 Mutate, 1 Swap
50	0.4348	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	1 Swap

Figure A.1: Abridged Best Chromosome History with Creating Operations, Part 1 of 3

Generation	Fitness	1	2	3	4	5	6	7	8	9	...	61	62	63	64	65	66	67	68	69	70	
51	0.4419	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	2 Swap
52	0.4679	9	15	13	10	16	17	16	11	2		2	3	4	5	1	6	1	12	3	1	2 Mutate, 2 Swap
54	0.4712	9	15	13	10	16	17	16	16	2		2	3	4	5	1	6	1	12	3	1	1 Mutate, 1 Swap
55	0.4723	9	15	13	10	16	17	16	16	2		2	3	4	4	1	6	1	12	3	1	1 Swap
58	0.4844	9	15	13	10	16	8	16	16	2		2	3	4	4	1	6	1	12	3	1	2 Swap
59	0.4959	9	15	13	10	16	17	16	16	2		2	3	4	4	1	6	1	12	3	1	2 Mutate
60	0.5062	9	15	13	10	16	17	16	16	2		2	3	4	4	1	6	1	12	3	1	2 Swap
64	0.5085	9	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate
69	0.5120	9	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate
72	0.5132	9	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Swap
73	0.5167	9	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate
76	0.5179	9	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	2 Mutate
81	0.5179	9	15	14	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	4 Mutate
84	0.5214	16	15	14	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate
85	0.5214	9	15	14	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate, 1 Swap
86	0.5249	16	15	14	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	1 Mutate
94	0.5260	16	15	13	10	16	17	16	16	2		2	2	4	4	1	6	1	12	3	1	3 Mutate
95	0.5284	16	15	13	10	16	17	16	16	2		2	2	4	4	1	4	1	12	3	1	1 Mutate
105	0.5307	16	15	13	10	16	17	16	16	2		2	2	4	4	1	4	1	12	3	1	2 Mutate
107	0.5330	16	15	13	10	16	17	16	16	11		2	2	4	4	1	4	1	12	3	1	1 Swap
111	0.5354	16	15	13	10	16	17	16	16	11		2	2	4	4	1	4	1	12	3	1	1 Mutate
112	0.5377	16	15	13	10	16	17	16	16	11		2	2	4	4	1	4	1	12	3	1	1 Swap
118	0.5389	16	15	13	10	16	17	16	16	11		2	2	4	4	1	4	1	12	3	1	1 Swap
124	0.5494	16	15	13	10	16	17	16	16	14		2	2	4	4	1	6	1	12	3	1	2 Mutate, 1 Swap
130	0.5551	16	15	13	10	16	17	16	16	14		2	2	4	4	1	6	1	12	3	1	2 Swap
134	0.5891	16	15	13	10	16	17	16	16	14		2	2	4	4	1	6	1	12	3	1	1 Mutate
140	0.5967	16	15	13	10	16	17	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Mutate, 1 Swap
143	0.5981	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
149	0.6062	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	2 Mutate
150	0.6062	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Mutate, 1 Swap
158	0.6110	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
161	0.6150	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
165	0.6193	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
174	0.6385	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
180	0.6400	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Mutate, 1 Swap
194	0.6440	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	2 Swap
222	0.6454	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
227	0.6516	17	15	13	10	16	16	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Mutate
241	0.6529	17	15	13	10	16	14	16	16	14		2	2	4	4	1	6	1	2	3	1	2 Mutate
250	0.6611	17	15	13	10	16	14	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
285	0.6625	17	15	13	10	16	14	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
314	0.6639	17	15	13	10	16	14	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
328	0.6651	17	15	13	10	16	14	16	16	14		2	2	4	4	1	6	1	2	3	1	1 Swap
344	0.6665	17	15	13	10	16	14	16	16	14		2	2	4	1	1	6	1	2	3	1	1 Swap
374	0.6720	17	15	13	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Mutate, 1 Swap
407	0.6734	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap
443	0.6736	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap
444	0.6750	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap
470	0.6778	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap

Figure A.2: Abridged Best Chromosome History with Creating Operations, Part 2 of 3

Generation	Fitness	1	2	3	4	5	6	7	8	9	…	61	62	63	64	65	66	67	68	69	70	
500	0.6819	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap
501	0.6833	17	15	14	10	16	14	16	16	14		2	2	4	1	1	1	1	2	3	1	1 Swap
505	0.6847	17	15	14	10	16	14	16	17	14		2	2	4	1	1	1	1	2	3	1	1 Mutate
542	0.6861	17	15	14	10	16	14	16	17	14		2	2	4	1	1	1	1	2	3	1	1 Swap
Reseeded at 742																						
747	0.6875	17	15	14	10	16	14	17	17	14		2	2	4	1	1	1	1	2	3	1	1 Swap
824	0.6875	17	15	14	10	16	14	17	17	14		2	2	4	1	1	1	1	2	3	1	1 Mutate
879	0.6881	17	15	14	10	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
976	0.6895	17	15	14	10	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
987	0.6915	17	15	14	10	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Mutate
990	0.6925	17	15	14	10	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1009	0.6934	17	15	14	12	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1032	0.6939	17	15	14	12	16	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Mutate
1067	0.6953	17	15	14	12	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1108	0.6962	17	15	14	12	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Mutate
1307	0.7018	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1354	0.7032	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1412	0.7065	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1423	0.7068	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1552	0.7082	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Mutate, 1 Swap
1634	0.7096	17	15	14	17	17	14	17	17	14		2	2	4	4	1	1	1	2	3	1	1 Swap
1775	0.7103	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
1836	0.7120	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
1855	0.7126	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
Reseeded at 2055																						
2089	0.7128	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2203	0.7131	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2297	0.7145	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	2 Swap
2403	0.7154	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2455	0.7168	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2580	0.7171	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2635	0.7185	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
2808	0.7198	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
Reseeded at 3108																						
3104	0.7213	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
3117	0.7220	17	15	14	17	17	14	17	17	14		2	1	4	4	1	1	2	2	3	1	1 Swap
Reseeded at 3317																						
Reseeded at 3517																						
Exited at 3717																						

Figure A.3: Abridged Best Chromosome History with Creating Operations, Part 3 of 3

APPENDIX B

Final Allocation Configuration: Individual Space Utilities

Table B.1: Final Allocation Configuration: Individual Space Utilities, Part 1 of 2

Space	Long. Pref	Vert. Pref	Min Global Pref	Relative Pref	Min Utility
1	1.0	1.0	1.0	0.7	0.7
2	0.6	1.0	0.6	0.6	0.6
3	0.6	1.0	0.6	0.6	0.6
4	1.0	1.0	1.0	0.7	0.7
5	1.0	1.0	1.0	0.7	0.7
6	0.6	1.0	0.6	0.6	0.6
7	1.0	1.0	1.0	0.7	0.7
8	1.0	1.0	1.0	0.7	0.7
9	0.6	1.0	0.6	0.6	0.6
10	0.6	1.0	0.6	0.6	0.6
11	1.0	0.4	0.4	0.4	0.4
12	1.0	0.4	0.4	0.4	0.4
13	1.0	0.2	0.2	0.3	0.2
14	1.0	0.4	0.4	0.4	0.4
15	1.0	0.4	0.4	0.4	0.4
16	1.0	1.0	1.0	0.8	0.8
17	1.0	1.0	1.0	0.7	0.7
18	1.0	1.0	1.0	0.8	0.8
19	1.0	1.0	1.0	0.5	0.5
20	1.0	1.0	1.0	0.8	0.8
21	1.0	1.0	1.0	0.8	0.8
22	1.0	0.7	0.7	0.7	0.7
23	1.0	1.0	1.0	0.8	0.8
24	1.0	1.0	1.0	0.8	0.8
25	1.0	1.0	1.0	0.8	0.8

Table B.2: Final Allocation Configuration: Individual Space Utilities, Part 2 of 2

Space	Long. Pref	Vert. Pref	Min Global Pref	Relative Pref	Min Utility
26	1.0	1.0	1.0	1.0	1.0
27	1.0	1.0	1.0	1.0	1.0
28	1.0	1.0	1.0	1.0	1.0
29	1.0	1.0	1.0	1.0	1.0
30	1.0	1.0	1.0	1.0	1.0
31	1.0	1.0	1.0	1.0	1.0
32	1.0	1.0	1.0	1.0	1.0
33	1.0	1.0	1.0	1.0	1.0
34	1.0	1.0	1.0	1.0	1.0
35	1.0	1.0	1.0	1.0	1.0
36	1.0	1.0	1.0	1.0	1.0
37	1.0	1.0	1.0	1.0	1.0
38	1.0	1.0	1.0	1.0	1.0
39	1.0	1.0	1.0	1.0	1.0
40	1.0	1.0	1.0	1.0	1.0
41	1.0	1.0	1.0	1.0	1.0
42	1.0	1.0	1.0	1.0	1.0
43	1.0	1.0	1.0	1.0	1.0
44	1.0	1.0	1.0	1.0	1.0
45	1.0	1.0	1.0	1.0	1.0
46	1.0	0.4	0.4	0.3	0.3
47	1.0	1.0	1.0	0.5	0.5
48	1.0	1.0	1.0	0.4	0.4
49	1.0	1.0	1.0	0.4	0.4
50	1.0	0.4	0.4	0.3	0.3
51	1.0	0.4	0.4	0.3	0.3
52	1.0	1.0	1.0	0.5	0.5
53	1.0	0.4	0.4	0.3	0.3
54	1.0	0.6	0.6	0.5	0.5
55	1.0	0.6	0.6	0.4	0.4
56	1.0	0.4	0.4	0.3	0.3
57	1.0	0.6	0.6	0.5	0.5
58	1.0	0.4	0.4	0.3	0.3
59	1.0	0.4	0.4	0.3	0.3
60	1.0	0.6	0.6	0.4	0.4
61	1.0	1.0	1.0	1.0	1.0
62	1.0	1.0	1.0	1.0	1.0
63	0.8	1.0	0.8	1.0	0.8
64	0.8	1.0	0.8	1.0	0.8
65	1.0	1.0	1.0	1.0	1.0
66	1.0	1.0	1.0	1.0	1.0
67	1.0	1.0	1.0	1.0	1.0
68	1.0	1.0	1.0	1.0	1.0
69	0.8	1.0	0.8	1.0	0.8
70	1.0	1.0	1.0	1.0	1.0
				Average Space Utility:	0.7271

BIBLIOGRAPHY

Bibliography

Ames, R. and R. Van Eseltine (2001). Architecture for multidiscipline integration of analyses in a common product model environment for LHA(R) topside. In *Electromagnetic Code Consortium (EMCC) Meeting*, Kauai, Hawaii.

Andrews, D. (1996, June). Subcon–A new Approach to Submarine Concept Design. *RINA WARSHIP 96 Symp. London*.

Andrews, D. (2003, September). A Creative Approach to Ship Architecture. *RINA International Journal of Maritime Engineering*.

Andrews, D. and C. Dicks (1997). The Building Block Design Methodology Applied to Advanced Naval Ship Design. *Proc. IMDC*, 3–19.

Andrews, D. and R. Pawling (2003). SURFCON - A 21st Century Ship Design Tool. *Proc. IMDC 3*.

Carlson, C. and D. Cebulski (1974). Computer-Aided Ship Arrangement Design. *Naval Engineers Journal 86*(5), 33–40.

Carlson, C. and H. Fireman (1987). General arrangement design computer system and methodology. *Naval Engineers Journal 99*(3), 261–273.

Cort, A. and W. Hills (1987). Space layout design using computer assisted methods. *Naval Engineers Journal 99*(3), 249–260.

Daniels, A. and M. Parsons (2006). An Agent-Based Approach to Space Allocation in General Arrangements. *Proc. IMDC*, 673–695.

Daniels, A. and M. Parsons (2007). Development of a hybrid agent-genetic algorithm approach to general arrangements. In *Proc. 5th International Conference on Computer Applications and Information Technology in the Marine Industries (COMPIT)*, Cortona, Italy, pp. 197–211.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY: Wiley.

Gen, M. and R. Cheng (1997). *Genetic Algorithms and Engineering Design*. New York, NY: Wiley Interscience.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts: Addison-Wesley.

Jo, J. and J. Gero (1998). Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering 12*, 149–162.

Kent, C. (2005). Original genetic algorithm code designed using materials from NA 570 Advanced Marine Design. Technical report, University of Michigan, Ann Arbor, MI.

Kosko, B. (1992). *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Lee, K., S. Han, and M. Roh (2002). Optimal compartment layout design for a naval ship using an improved genetic algorithm. *Marine Technology 39*(3), 159–169.

Li, J. and M. Parsons (1998). An Improved Method for Shipbuilding Market Modeling and Forecasting. *SNAME Transactions*, 157–183.

Li, J. and M. Parsons (2001). Complete design of fuzzy systems using a real-coded genetic algorithm with imbedded constraints. *Journal of Intelligent and Fuzzy Systems 10*(1), 13–37.

Medjdoub, B. and B. Yannou (2000). Separating topology and geometry in space planning. *Computer Aided Design 32*(1), 39–61.

Michalek, J. (2001). Interactive Layout Design Optimization. Master's thesis, University of Michigan.

Michalek, J., R. Choudhary, and P. Papalambros (2002). Architectural Layout Design Optimization. *Engineering Optimization 34*(5), 461–484.

Michalewicz, Z. (1996). *Genetic Algorithms+ Data Structures= Evolution Programs*. Springer.

Naval Sea Systems Command (1985). *NAVSEA Extended Work Breakdown Structure (ESWBS)* (59040-AA-IDX-010/SWBE 5D ed.). Washington, DC: Naval Sea Systems Command.

Naval Sea Systems Command (1992). *NAVSEA Design Practices and Criteria Manual for General Arrangements Design Chapter 070* (T9070-AB-PRO-010 Rev. A ed.). Washington, DC: Naval Sea Systems Command.

Naval Surface Warfare Center, Carderock Division (2005). *Advanced Surface Ship Evaluation Tool* (Version 5.2.0 ed.). West Bethesda, MD: Naval Surface Warfare Center, Carderock Division.

Nehrling, B. (1985). Fuzzy Set Theory and General Arrangement Design. *Proc. IFIP/IFAC Conf. on Computer Applications in the Automation of Shipyard Operations and Ship Design, Trieste*.

Nick, E. and M. Parsons (2007). Fuzzy Optimal Arrangement of Spaces within a Zone-deck Region of a Ship. *Proc. PRADS 1*, 666–673.

Nick, E., M. Parsons, and B. Nehrling (2006). Fuzzy Optimal Allocation of Spaces to Zone-decks in General Arrangements. *Proc. IMDC*, 651–671.

Ölçer, A., C. Tuzcu, and O. Turan (2006). An integrated multi-objective optimisation and fuzzy multi-attributive group decision-making technique for subdivision arrangement of Ro–Ro vessels. *Applied Soft Computing Journal 6*(3), 221–243.

Papalambros, P. and D. Wilde (2000). *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press.

Parsons, M., H. Chung, E. Nick, A. Daniels, S. Liu, and J. Patel (2008). Intelligent Ship Arrangements (ISA): A new approach to general arrangement. In *Proc. ASNE Day*.

Slapnicar, V. and I. Grubisic (2003). Multi - Criteria Optimisation Model of Deck Layout Design. *Proc. IMDC 3*.

Van Oers, B., D. Stapersma, and H. Hopman (2007). Development and implementation of an optimisation-based space allocation routine for the generation of feasible concept designs. In *Proc. 5th International Conference on Computer Applications and Information Technology in the Marine Industries (COMPIT)*, Cortona, Italy, pp. 171–185.