PROBLEM DECOMPOSITION BY MUTUAL INFORMATION AND FORCE-BASED CLUSTERING

A Thesis
Presented to
The Academic Faculty

by

Richard Edward Otero

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
May 2012

PROBLEM DECOMPOSITION BY MUTUAL INFORMATION AND FORCE-BASED CLUSTERING

To my wife

ACKNOWLEDGEMENTS

I come first to my wife Sara Kate Sams Otero. Thank you for your love, support, and good humor during this whole process. It has made a world of difference and has given me strength from a true partnership. Family has always been important to both of us and we have been blessed by two sets of parents we both adore. My love and thanks to my parents Mayra and Steven Hartofilis for raising a willful and clever child. Their love, kindness, and wisdom have helped make me the man I am today. My love as well to my newest parents Dana and Fletcher Sams who have made a home for me during this entire process in their life and hearts. Dana a wonderful mother and Fletcher a man I hope Justice emulates. With step-siblings, we now have three sisters and two brothers. I could not ask for better.

Robert Braun has been an inspiring advisor and teacher. He is a large part of where I find myself professionally and where I will be in the future. I thank him for his support, humor and the choice he made in accepting me into the Space Systems Design Lab (SSDL). To the rest of my thesis committee, Ian Clark was able to co-advise my work during Bobby's tenure at NASA. Ian's insights, advice and rigorous editing have been deeply appreciated. Charles Isbell provided my first introduction to Machine Learning and the computational mechanics involved with Artificial Intelligence. Charles mixes his subjects with a joyful repartee and has been a friend. My thanks go also to John-Paul Clarke for his feedback and for his ever heartening presence. George Chen has shown great willingness to help in my endeavors and has greatly supported the development of the Planetary Entry Systems Synthesis Tool (PESST) that has been developed as part of this thesis. I look forward to working with George at JPL. The entire committee has my heartfelt thanks.

The SSDL has served as a lifeline for my desire to work on space systems. The caliber of the men and women I have met at the lab has been extraordinary and I hope to keep contact with them into the future. I particularly want to thank Michael Grant, Bradley Steinfeldt and Patrick Smith for their friendship and work on PESST. Many thanks to Gregory Lantoine for the low-thrust trajectory talks and to Nitin Arora for our conversations on the ever increasing potential of GPUs. There are many other friends that I have been honored to know during my time at Georgia Tech; thank you for exploring the absurdities of life with me.

When I first selected Systems Design, I joined the Aerospace Systems Design Lab under Dimitri Mavris. I thank him for making a place for me in his lab while providing the freedom necessary to take the space design qualifying exam.

I would like to thank the Jet Propulsion Laboratory and the Charles Stark Draper Laboratory for their sponsorship of the new PESST framework; continuing through the efforts of Ian Meginnis and Zack Putnum. I am glad for those who have found continuing use for the framework and for those who will continue development efforts. My thanks to the GEM Fellowship program which enabled my first masters degree. They are a wonderful program supporting the efforts of minority students seeking degrees in engineering and science. Stephen Ruffin, a wonderful teacher who deeply changed my understanding of gases and computational fluid dynamics, made the recommendation that led to my Alfred P. Sloan Foundation Graduate Scholarship. I thank him for his support and the support of the Alfred P. Sloan Foundation. The National Institute of Computational Sciences partnered with Georgia Tech, Oak Ridge National Laboratory, NVIDIA and the University of Tennessee to create the Keeneland cluster. Time on the Keeneland cluster was generously donated to enable the studies performed in this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

a	Acceleration magnitude.
A	Area.
a_{ts}	Tauber and Sutton exponential constant.
AZ	Azimuth.
B	Material constant in Arrhenius relation.
b_{ts}	Tauber and Sutton exponential constant.
BC	Ballistic Coefficient.
C	Constant.
C_A	Axial-Force Coefficient.
C_D	Force Coefficient of Drag.
C_L	Force Coefficient of Lift.
C_N	Normal-Force Coefficient.
C_P	Pressure Coefficient or Specific heat of TPS material.
C_X	Force Coefficient along x-axis.
C_Z	Force Coefficient along z-axis.
C_k	Constant used for DGB parachute sizing.
CAD	Computer-aided design.
CH_4	Methane.
CPU	Central Processing Unit.
$CUDA$	Compute Unified Device Architecture.
D	Drag.
D_o	Parachute diameter.
DOF	Degree-of-Freedom.
DSM	Design Structure Matrix.
E	Material constant in Arrhenius relation.

EDL	Entry, Descent and Landing.
F	Force.
$f()$	Function of.
f_{safety}	Safety factor.
$FM5055$	A type of Carbon Phenolic.
FPA	Flight path angle.
g	Gravitational acceleration magnitude.
g_E	Unit of Earth gravity.
g_o	Acceleration of Gravity on the Earth's Surface.
$GA(s)$	Genetic Algorithm(s).
GNU	GNU's Not Unix (recursive acronym).
GPU	Graphical Processing Unit.
$GRAM$	Global Reference Atmosphere Model.
GSE	Global Sensitivity Equations.
GTS	GNU Triangulated Surface.
GUI	Graphical User Interface.
H	Enthalpy.
H_{ref}	Reference Enthalpy at 298K.
i	An index for the column number.
I	Identity matrix.
I_{SP}	Specific Impulse.
iid	independent and identically distributed.
j	An index for the row number.
J	Performance index.
JMP	Statistical Software (originally John's Macintosh Project).
k	Thermal conductivity.
k_{sg}	Sutton-Graves convective heating constant.

$KISS$	Keep It Simple Stupid.
KL	Kullback-Leibler.
L	Lift magnitude.
$LGPL$	GNU Lesser General Public License.
LOX	Liquid Oxygen.
m	mass.
$MATLAB$	Matrix Laboratory.
$MDAO$	Multidisciplinary Design Analysis and Optimization.
$MIMIC$	Mutual Information Maximizing Input Clustering.
MMH	Monomethyl Hydrazine.
MSI	Module Strength Indicator.
n	Magnitude of normal vector.
NN	Neural Networks.
NP	Non-deterministic Polynomial Time.
$NVIDIA$	Producer of graphics cards.
OBD	Optimizer-Based Decomposition.
P	Pressure.
p	probability.
p_{approx}	Approximation to the true joint probability distribution.
p_{true}	True joint probability distribution.
$PESST$	Planetary Entry Systems Synthesis Tool.
$PICA$	Phenolic Impregnated Carbon Ablator.
q	Dynamic pressure or Stagnation point heating.
Q^*	Thermochemical heat of ablation.
R	Radius.
r	Position magnitude.
R_{ug}	Universal gas constant.

RSE	Response Surface Equation.
s	Thickness of material lost from ablation.
$SLA-561v$	Super Light weight Ablator.
STL	Stereolithography.
T	Thrust or Temperature.
t	Time.
TPS	Thermal Protection System.
v	Velocity magnitude.
V	Volume.
var	Stand in for any other variable.
W	Weight.
x, y, z	Axial directions or unspecified values.
X, Y, Z	Random variables.
	Greek Symbols.
α	Angle of attack.
Δ	Change in.
ϵ	Density ratio across shock wave.
η	Transpiration coefficient.
γ	Ratio of specific heats.
Γ	Importance weighting on the final time.
Γ_{resin}	Resin volume fraction in TPS material.
ϕ	Flight path angle.
ϕ_{tank}	Mass factor describing the strength of tank material.
ρ	Density.
σ_{SB}	Stephan-Boltzman constant.
θ	Angle between surface and oncoming flow.
ε	Emissivity.

Subscripts.

$var_{A,B,C}$	Identifiers to signify different items.
var_{AS}	After shock.
var_{BS}	Before shock.
var_{∞}	Value in the free stream.
var_{air}	of air.
var_{burst}	Value at burst.
var_{conv}	Convective.
var_{cw}	Cold wall.
var_{design}	Value used in design.
var_{entry}	Regarding value at entry.
var_{fiber}	Fiber in TPS material.
var_{f}	Final.
var_{go}	To-go.
var_{max}	Maximum value.
var_{mortar}	Parachute mortar.
var_{n}	Effective nose.
var_{o}	Initial.
var_{panel}	With respect to panel.
var_{para}	Parachute.
var_{rad}	radiative.
var_{ref}	Reference.
var_{resin}	Resin in TPS material.
var_{r}	recovery.
var_{struct}	Structure.
var_{tank}	Regarding fuel tank.
var_{tot}	Total.

var_v	of vaporization.
var_w	wall.
var_x	Along x-axis.
var_y	Along y-axis.
var_z	Along z-axis.
	Superscripts.
$v\bar{a}r$	Vector.
$v\dot{a}r$	Time derivative.
var^I	Inertial frame.
var^{Tw}	Evaluated at the wall temperature.
var^T	Transpose.
var^{rel}	Planet relative.

SUMMARY

The scale of engineering problems has sharply increased over the last twenty years. Larger coupled systems, increasing complexity, and limited resources create a need for methods that automatically decompose problems into manageable sub-problems by discovering and leveraging problem structure. The ability to learn the coupling (inter-dependence) structure and reorganize the original problem could lead to large reductions in the time to analyze complex problems. Such decomposition methods could also provide engineering insight on the fundamental physics driving problem solution.

This work forwards the current state of the art in engineering decomposition through the application of techniques originally developed within computer science and information theory. The work describes the current state of automatic problem decomposition in engineering and utilizes several promising ideas to advance the state of the practice.

Mutual information is a novel metric for data dependence and works on both continuous and discrete data. Mutual information can measure both the linear and non-linear dependence between variables without the limitations of linear dependence measured through covariance. Mutual information is also able to handle data that does not have derivative information, unlike other metrics that require it. The value of mutual information to engineering design work is demonstrated on a planetary entry problem. This study utilizes a novel tool developed in this work for planetary entry system synthesis.

A graphical method, force-based clustering, is used to discover related sub-graph structure as a function of problem structure and links ranked by their mutual information. This method does not require the stochastic use of neural networks and could be used with any link ranking method currently utilized in the field. Application of this method is demonstrated on a large, coupled low-thrust trajectory problem.

Mutual information also serves as the basis for an alternative global optimizer, called MIMIC, which is unrelated to Genetic Algorithms. Advancement to the current practice demonstrates the use of MIMIC as a global method that explicitly models problem structure with mutual information, providing an alternate method for globally searching multi-modal domains. By leveraging discovered problem interdependencies, MIMIC may be appropriate for highly coupled problems or those with large function evaluation cost. This work introduces a useful addition to the MIMIC algorithm that enables its use on continuous input variables. By leveraging automatic decision tree generation methods from Machine Learning and a set of randomly generated test problems, decision trees for which method to apply are also created, quantifying decomposition performance over a large region of the design space.

CHAPTER I

MOTIVATION, BACKGROUND, AND STUDY OBJECTIVES

1.1 Motivation

The scale of multidisciplinary problems in engineering has greatly increased over the past twenty years; twenty design variables once was typical of a large scale conceptual problem while now this number can exceed a hundred.[1] The processing power of computers has roughly followed the prediction of Moore for over thirty years, doubling at 1.5 year intervals for a computer of the same cost. Computational fluid dynamics problems that once were examined in doctoral dissertations are now routinely assigned as homework problems. Yet the fidelity and complexity of examined engineering design problems has kept track with the rapid rise in processing capability. Higher fidelity tools are being used in earlier stages of the design process and larger problem domains are being examined. This is to say that the problems tackled by engineers continue to evolve towards higher fidelity and wider domain exploration, always on the edge of current processing capability. Researchers today are presented with the choice of either waiting for the processing power to deal with their harder problems or developing methods to handle these problems with today's hardware.

Decomposition methods expand the problems currently solvable by today's processing capabilities. The philosophy of 'divide and conquer' has known as much use within the battlefield and scientific thought as within the arts. A larger problem is decomposed into smaller, more tractable, sub-problems of appropriate structure. The solutions to these smaller problems are then used to either solve or provide insight into the behavior of the original problem. Historically this process has been performed by human discipline experts and many heuristics have been developed to guide these decomposition efforts. [42, 79, 80, 100, 103] The problem has always existed of what to do when either a professional is not available or when a problem does not agree with an individual's experiences or expectations.

Large highly coupled problems will become more the norm in the future as higher fidelity tools move to earlier in the design process and more ambitious problems are considered in engineering. The complexity of variable interactions between coupled analyses will provide challenges to human discipline experts seeking to decompose larger problems into more tractable components. It is proposed that this creates a need for automated decomposition methods to aid the engineer in discovering useful sub-problem structure that can be leveraged to understand and better solve their problems.

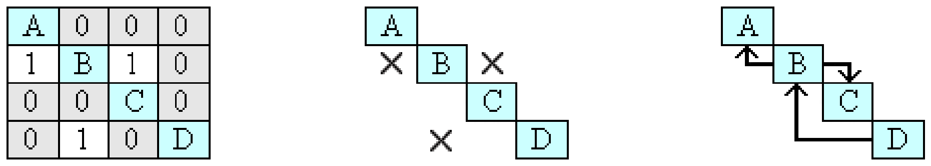
The ability to automatically learn the coupling (interdependence) between disciplines could lead to large reductions in the time to analyze problems.[50] This time reduction comes by reducing the combinatorial design space examined for a solution. Such decomposition methods could also provide insight to an engineer on the physics driving a problem. This work aims to forward the current state of the art in engineering decomposition.

1.2 Problem Representation

The most common problem representation used in multidisciplinary design analysis and optimization (MDAO) for engineering design is the design structure matrix (DSM) or N^2 diagram.[15] The design structure matrix was introduced by Steward[91] as a tool for marking the interactions between analyses.

The analyses can be any programs or modules that accept zero or more inputs and provide zero or more outputs. Information output from one analysis can serve as inputs into another. The ordering of the DSM diagonal from top left to bottom right typically provides the general execution ordering for the modules, though ready modules can be run concurrently. When an analysis provides data to a point lower in the DSM diagonal, i.e. to an analysis planned by the DSM to run later, it is called a feed-forward link. When data is passed to a point higher in the DSM diagonal it is called a feedback link. Three equivalent graphical representations of a DSM are provided in Figure 1.

A design structure matrix is commonly discussed in engineering classrooms and is used by most engineering decomposition methods for its ease of implementation. A DSM is useful as a means to clarify the interactions between analyses. This paper will follow the DSM format where feedback connections are below the diagonal and feed-forward connections are shown above the diagonal, Figure 1.

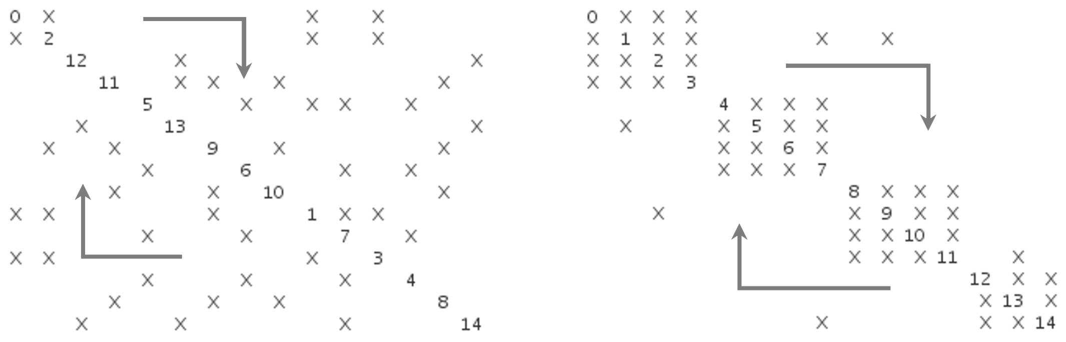


(a) Linked by Weight (b) Linked by Marker (c) Linked by Arrows

Figure 1: Three Equivalent Graphical Representations for a DSM.

This work will primarily use the representation shown in Figure 1(a) as additional information regarding the strength of the link can be shown in the graph. When the diagram is used to only display the existence of a link, Figure 1(b) will be used for clarity. Regardless of the representation used, links above the diagonal show the forward passing of data to later analyses and links below the diagonal show the feed-back of information to earlier analyses, shown by arrows in Figure 1(c).

A decomposed (well ordered) DSM better shows the structure of the problem. Strongly connected analyses can be easily seen and links between analysis clusters can be evaluated to see if clusters can be treated as separable sub-problems. A randomly ordered DSM is shown in Figure 2(a). The analysis programs have been randomly ordered along the diagonal of the graph and engineers examining the graph would have difficulty finding problem structure to leverage towards decomposing the problem. The exact same problem is shown in Figure 2(b) with an ordering that helps show possible decompositions for the problem. Four separate closely connected groups might be separable enough to be considered as sub-problems. This decision would depend on the strength of links connecting these sub-groups.

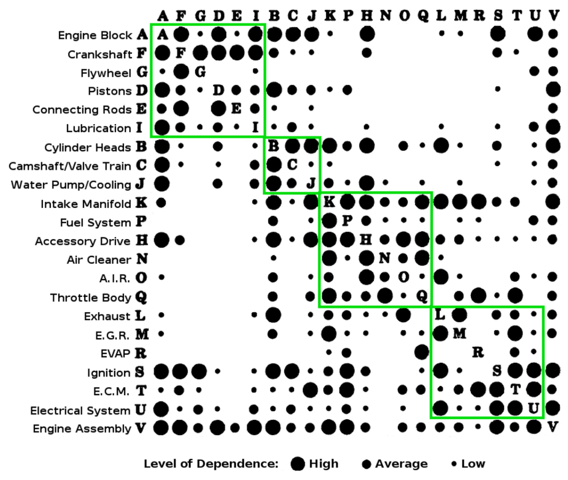


(a) Randomly Ordered DSM (b) Decomposed DSM

Figure 2: Two Orderings for the same DSM Problem Structure

Larger coupled problems will experience the same difficulty as Figure 2(a), i.e. the lack of easily displayed sub-structure. A DSM likely will not begin with an ordering that makes plain potential sub-structures within the problem. For instance, General Motors once sought to organize engine sub-system groups into larger composite groups that could work as independently as possible from other composite groups. Their original manual attempt at decomposition is shown in Figure 3(a). This attempt at grouping sub-system groups manually found a set of groupings that still required a strong amount of communication across the larger groups.[62]

A study was made to rank the frequency of interactions between sub-system groups as a heuristic for interaction importance. This ranking was performed by questioning management, manufacturing engineers and automotive engineers through surveys where responses for rank could easily differ.[62] A combination of conservative and averaged estimates for rank weighted the interactions of the DSM. Group clusters


(a) Manually arranged DSM

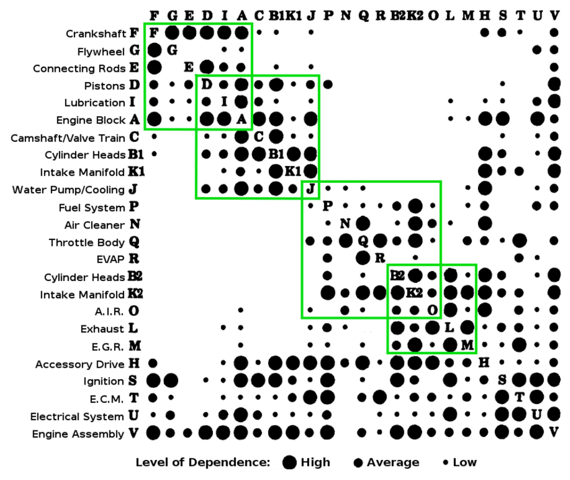

(b) Computationally arranged DSM

Figure 3: Two DSM Orderings for a General Motors Engine[62]

with strong requirements for inter-communication were placed into larger composite groups by computationally rearranging the DSM. The algorithm used in the study highlighted two heavily connected sub-systems (B and K in Figure 3(b)) that could open better decoupling by being replicated. This and computational rearrangement allowed for groups with better interactions, Figure 3(b). This example, while relatively small, was still challenging to manually expose the sub-structure that was later found through automated decomposition.

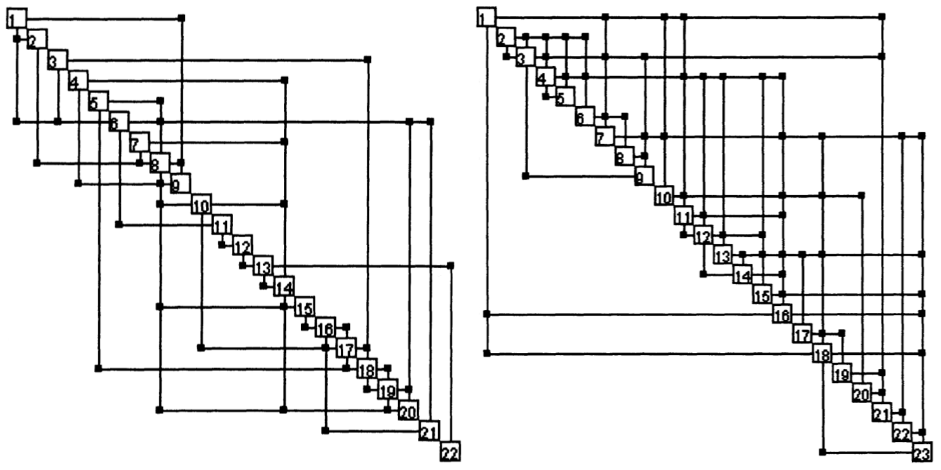


Figure 4: Two DSM Examples of Aerospace Conceptual Design Problems[81]

Two aerospace conceptual design problems were shown by Rogers, et al.[81] through work performed at Langley on automated decomposition methods. Both conceptual design problems are considered in this work to be of medium size and complexity. The DSM in Figure 4(a) shows roughly 8% of its links active while Figure 4(b) has over 11% of its links active. This work will examine both smaller and larger problems, and examine complex problems with 20% of their links active.

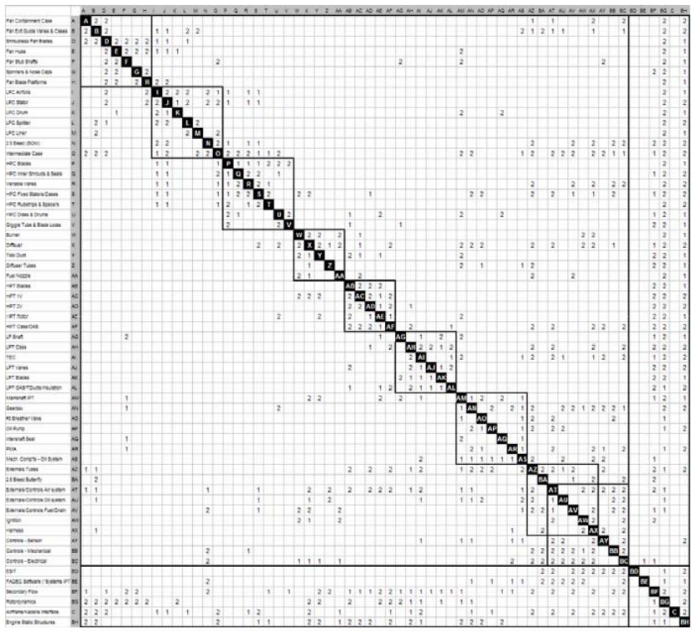
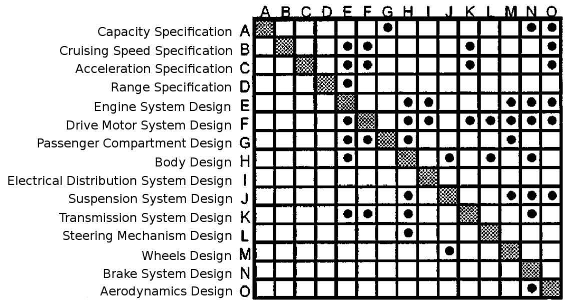


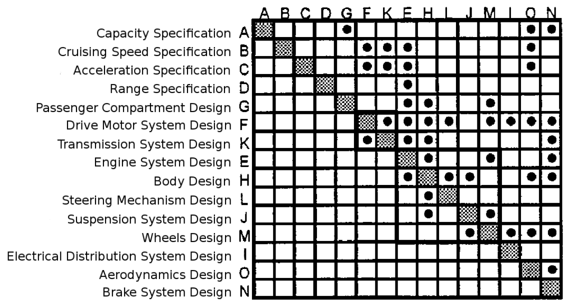
Figure 5: Pratt and Whitney Manually Arranged DSM for Turbofan Engine[104]

A large DSM example with 60 analyses is shown by Yu, et al.[104] in Figure 5 for a Pratt and Whitney Turbofan Engine. This DSM was manually decomposed with information links that were ranked based on collected surveys. This type of aerospace problem, where manual methods become unpractical, would benefit well from automated decomposition methods.

The problem of sub-optimum DSM arrangement exists when a designer is scheduling the execution ordering for design tasks. In Figure 6, the design process for an automobile is reordered to remove a number of feedback links that would each require an analysis module to be re-executed. Ideally, with no feedback links, every analysis block would only have to be executed once.[15]


A B C D E F G H I J K L M N O
Capacity Specification A
Cruising Speed Specification B
Acceleration Specification C
Range Specification D
Engine System Design E
Drive Motor System Design F
Passenger Compartment Design G
Body Design H
Electrical Distribution System Design I
Suspension System Design J
Transmission System Design K
Steering Mechanism Design L
Wheels Design M
Brake System Design N
Aerodynamics Design O

(a) Before Rearrangement


A B C D G F K E H L J M I O N
Capacity Specification A
Cruising Speed Specification B
Acceleration Specification C
Range Specification D
Passenger Compartment Design G
Drive Motor System Design F
Transmission System Design K
Engine System Design E
Body Design H
Steering Mechanism Design L
Suspension System Design J
Wheels Design M
Electrical Distribution System Design I
Aerodynamics Design O
Brake System Design N

(b) After Rearrangement

Figure 6: Two DSM Orderings for Automobile Design Process[15]

The next section will examine several heuristics that have been used in engineering to reorder a DSM into a structure that is more constructive to informing about problem sub-structure. These heuristics are normally based on expert input and/or values that can be calculated without running any of the analyses in the DSM (i.e. a static analysis). In Section 2.3, a static method based on the information theoretic construct of mutual information will be explained to advance the current state of the art for reordering DSMs into the more ideal structure seen in Figure 2(b), Figure 3(b) or Figure 6(b).

A distinction will be made here between DSMs that are used in the literature, based on the form of their linkage information. As seen in Figure 7, the three distinct groups are: binary, discrete, and real value weighted DSMs.

A	0	0	0
1	B	1	0
0	0	C	0
0	1	0	D

(a) Weighted (Binary)

A	0	0	0
2	B	3	0
0	0	C	0
0	5	0	D

(b) Weighted (Discrete)

A	0	0	0
1.4	B	2.6	0
0	0	C	0
0	7.1	0	D

(c) Weighted (Real)

Figure 7: Types of Design Structure Matrices.

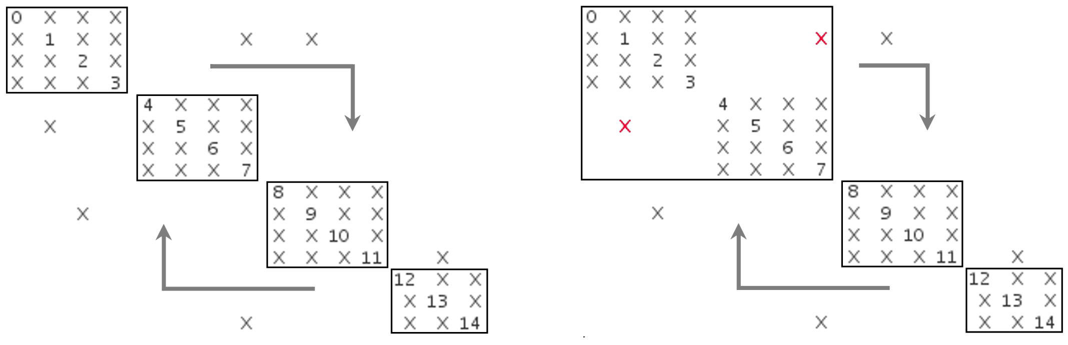
A binary DSM shows data dependence without information on the weight or importance of the information link. A weighted DSM provides either a discrete or real valued importance to the information link between analyses. Heuristics incorporating expert opinion or the number of variables passed (also referred to as the 'thickness of the pipe') typically provide discrete values (1,2,3 or low/med/high, etc). Gradient based information or other computed metrics often use real values to rank link importance.

1.3 Methods for Decomposition

1.3.1 Link Importance Discovery

When evaluating how to decompose a coupled problem into sub-problems, a great challenge has been the correct evaluation of the importance due to each of the interconnections. A highly coupled problem with unimportant connections can be approximated as a decoupled problem (without interconnections); the connections being re-added only to refine the final answer.[91] On the other hand, the mislabeling of a highly important interaction would damage the assumption used to decompose the problem in this way. A coupled problem, with important interconnected dependencies, may challenge efforts to separate it into smaller tractable pieces. Researchers have hoped to leverage knowledge of dependencies to gain the benefits of decomposition.[15]

The four closely connected groups are considered as sub-problems in Figure 8(a). This temporarily assumes that the links between the groups are unimportant enough to disconnect while four sub-solutions are found. The four groups could then be reconnected to converge onto a system solution with good initial guesses for the sub-solutions of the problem. If the links connecting the first two sub-groups in Figure 8(b) are in fact very important towards determining their sub-solutions it might be better to treat the problem as having three sub-problems. Here, the structure for the sub-problems is based on the inter-dependence of the links. A useful dependence metric from information theory, mutual information, will be described in Section 2.1 along with its advantages over several current link importance heuristics.



(a) Every Group formed into Sub-problems

(b) Composite Sub-problem formed

Figure 8: Two Potential Sub-problem Organizations for the same DSM

Most methods have dealt with decomposing the problem before the running of an analysis (static decomposition) by: treating all links as equally important[16, 103] (binary connections), using heuristics such as the number of variables passed through the link[1], and/or by using survey results from experts to rank linkage importance with a discrete value (e.g. low/med/high).[80] Weighted DSMs can also have real values assigned to the links. These values have come from sensitivity calculations[79] or other metrics used to calculate a real valued importance for the link.[42, 17]

Global Sensitivity Equations have been used at run-time to define the total derivatives of the output responses in terms of the subsystem local sensitivities.[79] This is an instance where the dynamic importance of the links, given by the output response's sensitivity to changes, is computed at run-time. The method requires that derivatives can be taken at both the subsystem and global level, but is one of the few methods that allow the problem behavior to dictate the importance that should be assigned to interdependencies.[88] This has, for instance, seen use in the multidisciplinary synthesis of aircraft.[33]

Future problems will tend to be larger and more complicated following increases to analysis and system fidelity. The ability to explicitly rank the interconnections between analyses will aid in the real time decomposition of problems as they are evaluated. Without advancing the methods currently used for decomposition, problem tractability will primarily depend on currently available hardware. Though the advancement of computational power is impressive, the opportunity of pairing this power with real-time (or dynamic) decomposition methods should allow the efficient solution of larger-scale problems.

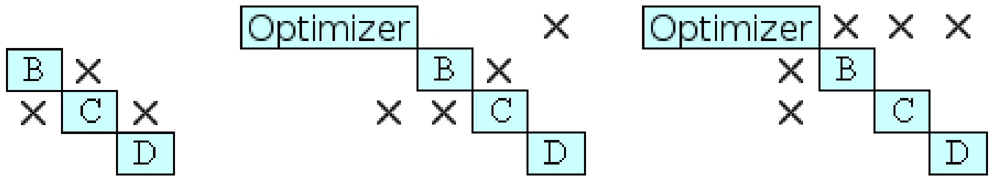
1.3.2 Static Decomposition: Pre-Execution Methods

Decomposition methods can be classified by when they act to decompose the problem being examined. Static decomposition methods seek to separate a problem into subproblems before executing any of the component analysis programs. When this is done, the overhead in decomposing the problem is front loaded so the user does not have to pay that cost during execution. Expert knowledge can be used to rank the importance of links in a DSM, without running any of the analyses, and these rankings can be leveraged to form clusters of related analyses. A potential drawback with this pre-execution arrangement is that the expert opinion and/or heuristics are assumed applicable to the particular problem being analyzed.

1.3.2.1 Optimizer-Based Decomposition

One current engineering method for pre-execution decomposition replaces several links in the design matrix with new constraints that are controlled by an added optimizer.[10, 28, 97] The analyses can be run without the converted links and the optimizer seeks to assure that any answer also satisfies an added set of compatibility constraints.[11, 13] These constraints ensure that, at convergence, variable values agree between their source and destination analysis blocks.

This method has been used to remove a subset of links in an analysis (partial-OBD) or to remove all of the links within an analysis (full-OBD), see Figure 9.[69]



(a) Original DSM (b) Partial-OBD to Remove Link (c) Full-OBD

Figure 9: Examples of Optimizer Based Decomposition.

Partial-OBD can be used to remove analysis iteration loops from a design problem while full-OBD would allow each analysis the ability to run concurrently. A drawback to this method has been that converging each new constraint might take a large number of iterations; fewer if the former links are well behaved or if their value does not change the tracked output greatly.[14]

1.3.2.2 Optimizer-Based Heuristic Decomposition

Heuristics continue to be widely used to reorder the analyses in a DSM before the running of any of the processes.[42, 79, 80, 100, 103] Table 1 is a representative list of the heuristics that have been used to automatically decompose problems in engineering.

The list includes several examples of pre-execution decomposition and some later entries that begin to incorporate during-execution information into a decomposition (dynamic decomposition). When pre-execution decomposition is used, it utilizes static heuristics as a stand in for explicitly computing the importance for the information links. The incorporation of during-execution data aims to utilize discovered information to better estimate link placement and importance.

Table 1: Overview of Published Utility Functions used for Problem Decomposition. (Provided in Chronological Order)

Decomp. Method	Problem Repres.	Optim. Method	Utility Metric Description	Utility Function	Reference
Pre-execution	Binary DSM	Basic GA	Minimize the number of feedback connections; aims to reduce iterations.	min $f = \sum_{i=1}^{n} \sum_{j=i+1}^{n} v(i,j)$ where $v(i,j)$ is either 0 or 1.[1]	Steward (1981) [91]
Pre/During-execution mix based on GSE Sensitivities	Real value weighted DSM	Rule based logic	Global Sensitivity Equations used to define the total derivatives of the output responses in terms of the subsystem local sensitivities	Normalized local sensitivities; local behavior is assumed as differentiable.	Rogers and Bloebaum [DeMAID] (1994) [79]
Pre/During-execution mix based on GSE Sensitivities, Static Loop Heuristics	Weighted DSM	Basic GA	GSE generated strengths and a user generated prediction as to how often a loop will iterate. Two iterations assumed for very weak feedbacks; 8 for very strong feedbacks.	Given estimates for each analysis cost, attempts to find the least costly configuration.	Rogers [DeMAID-GA] (1996) [78]

Table 1: Overview of Published Utility Functions used for Problem Decomposition. (Provided in Chronological Order)

Decomp. Method	Problem Repres.	Optim. Method	Utility Metric Description	Utility Function	Reference
Pre-execution	Discrete DSM	GA with permutation based operators	Links in matrix weighted by the number of variables passed through each linkage (thickness of pipe). Each variable seen as equally important. Considered bandwidth concerns and database size.	Objective used to compare against DeMAID for 'total length of feedback' was $f = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (j-i) w(i,j)$ where $w(i,j)$ is that number of variables present in the link (i,j).[1]	Altus et al. [AGENDA] (1996) [1]
Pre-execution	Binary DSM	Basic GA	Summed distance of 1's from left of matrix Summed distance of 1's from bottom of matrix (concurrency) Summed distance of 1's below diagonal (reduce feedback)	min $f = \sum_{i=1}^{n} \sum_{j=1}^{n} (i) v(i,j)$ min $f = \sum_{i=1}^{n} \sum_{j=1}^{n} (n-j) v(i,j)$ min $f = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (j-i) v(i,j)$ where $v(i,j)$ is either 0 or 1 (showing link existence)[1]	Todd (1997) [95]
Pre-execution	Discrete DSM	Basic GA	Heuristic to maximize the number of internal module dependencies while seeking to minimize inter-module dependencies by using a Module Strength Indicator (MSI)	$MSI = MSI_i - MSI_e$ $MSI_i = \frac{\sum_{i=n_1}^{n_2} \sum_{j=n_1}^{n_2} w_{i,j}}{(n_2-n_1)^2-(n_2-n_1)}$ $MSI_e = \frac{\sum_{i=0}^{n_1} \sum_{j=n_1}^{n_2} w_{i,j}+w_{j,i}}{2n_1(n_2-n_1)} + \frac{\sum_{i=n_2}^{N} \sum_{j=n_1}^{n_2} w_{i,j}+w_{j,i}}{2(N-n_2)(n_2-n_1)}$	Whitfield et al. (2002) [100]

Table 1: Overview of Published Utility Functions used for Problem Decomposition. (Provided in Chronological Order)

Decomp. Method	Problem Repres.	Optim. Method	Utility Metric Description	Utility Function	Reference
Pre-execution	Discrete DSM	Basic GA	Heuristics for tasks where a task can occur in parallel to a later task, often seen during scheduling. Shows a method to use rank information, if it is present, but not how to find it.	Minimize feedbacks, group tasks towards diagonal, tasks that can execute a section by themselves should do so.	Whitfield et al. (2005) [99]
During-execution	Neural Net-works	Genetic operations	Seeks to model subsets of data with NNs evolved by GAs. Time intensive but very flexible representation. A population of neural networks is created with randomly selected inputs.	Well performing networks are kept and used to create new population. A separate population selects for a combining network, pulling NN modules from the first population.	Khare (2006) [46]
During-execution	Discrete DSM	Basic GA	The use of an information theoretic measure, Minimum Description Length	Among all models, choose the one that uses the min length for describing a given data set well.	Yu et al. (2007) [105]

[1]The index i refers to the column number while j refers to the row index.

1.3.3 Dynamic Decomposition: During-Execution Methods

Dynamic decomposition seeks to decompose a problem with information gained during solution of part or all of the original problem. In learning the unique behavior of a problem, the first issue is often how to separate the problem into sub-problems. Three methods that are often used in engineering are examined here: derivative-based methods, neural networks and the genetic crossover operation. Both neural networks and genetic crossover seek to find the underlying structure of a problem without a focus on providing that structure in a human-readable format. The methods provide answers to their users without an explanatory focus, either stochastically (genetic algorithms) or through back propagation from a large training set (neural networks). In contrast, derivative-based methods generally provide an explanation to their ranking of information links that is more understandable to the engineer as a table of partial derivatives contains structural information on the problem. A list of network node weights from a neural network graph does not have the same explanatory power as a table of partials. This work does not address dynamic decomposition directly but does support the future development of dynamic methods by the introduction of a flexible link importance metric and a graphical force-based clustering method.

1.3.3.1 Derivative-Based Sensitivity Methods

Global Sensitivity Equations (GSE) are often used in engineering and allow for the approximation of link importance through the discovery of system analysis sensitivities to changes in the design variables.[88] An input vector $\mathbf{x}$ of design variables are passed into an analysis which produces a vector of state variables $\mathbf{y}$. The computed sensitivity $\frac{\mathrm{d}\mathbf{y}_a}{\mathrm{d}x}$ represents the change in the state variables from analysis A due to a change in the design variables from $\mathbf{x}$.[70]

$$\begin{bmatrix} I & -\frac{\partial \mathbf{y}_a}{\partial \mathbf{y}_b} & -\frac{\partial \mathbf{y}_a}{\partial \mathbf{y}_c} \\ -\frac{\partial \mathbf{y}_b}{\partial \mathbf{y}_a} & I & -\frac{\partial \mathbf{y}_b}{\partial \mathbf{y}_c} \\ -\frac{\partial \mathbf{y}_c}{\partial \mathbf{y}_a} & -\frac{\partial \mathbf{y}_c}{\partial \mathbf{y}_b} & I \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}\mathbf{y}_a}{\mathrm{d}x_j} \\ \frac{\mathrm{d}\mathbf{y}_b}{\mathrm{d}x_j} \\ \frac{\mathrm{d}\mathbf{y}_c}{\mathrm{d}x_j} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{y}_a}{\partial x_j} \\ \frac{\partial \mathbf{y}_b}{\partial x_j} \\ \frac{\partial \mathbf{y}_c}{\partial x_j} \end{bmatrix} \tag{1}$$

The sensitivity strength of the analysis state variables to the design variables serves to approximate the importance for the data links for this analysis. The derivatives can be taken either analytically, when possible, or by using finite differences. This assumes that the variables are either continuous or are discretized versions of continuous variables. Naturally discrete variables (e.g. engineA, engineB, engineC) could not be addressed by this method. Ford and Bloebaum [26] developed an algorithm to optimize mixed systems by separating discrete and continuous spaces and treating the continuous spaces with GSE. It is important to note that the derivatives refer to the current point in the design space. Moving to a different section of the design space would generally have different partials.

A decomposition could then be performed by using an optimizer to find a useful rearrangement of analyses given the ranked linkage information. A ranked list of links provides suggestions to the reorganization problem with the ability to more freely manipulate links with low sensitivities to changes in the design variables.

1.3.3.2 The Genetic Crossover Operation

Genetic Algorithms (GAs) are currently an accepted method for searching large multimodal domains that have inexpensive fitness function calls.[38] The requirement regarding the execution time for the fitness function is needed as GAs often require several hundred thousand function calls to converge to a solution. A GA attempts to model the higher performing areas of the solution space by maintaining a population of high performing candidate solutions. The biologically inspired operations of reproduction, crossover and mutation are performed on these solutions to potentially improve their utility over successive generations (iterations).[30]

The representation and genetic operation of crossover present challenges to a genetic algorithm's indirect modeling of variable interactions. These challenges are addressed in Section 2.3 by the introduction and extension of a method from Computer Science.

A candidate solution in genetic algorithms takes the form of an array of values. In Figure 10, the binary string provides values for eight binary input variables. This potential solution gives one combination of input values for the problem and will later be assigned a measure of worth based on a user provided utility function.[38] The operations in genetic algorithms aim to combine segments from high performing candidate solutions together to hopefully provide better child solution strings. The assumption made is that high performing solutions have solved an aspect to the problem that can be passed on and combined with other sub-solutions to create a better performing 'child'. This is sometimes known as the Building Block Hypothesis.[31]

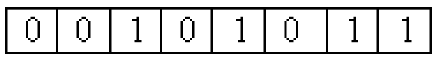


Figure 10: Example GA Binary String.

Input-variable dependencies, for a solution space, provide information on how inputs might be separated into different sub-problems. This dependency information is modeled indirectly by GAs. The chromosome representation for GAs and the most common implementations of the crossover operator assume that coupled variables appear near each other on the candidate string. This bias of the crossover operation, will be shown through the examples of this section.

The application of crossover within a GA randomly separates out a sub-solution, to an unknown sub-problem, from one parent and combine it with a sub-solution from a second parent. The child, so created, would therefore have the incorporated solutions developed from each parent and ideally perform better than either parent.[82] GAs do not compute the explicit relationships between input variables to determine where the best crossover points should be. They randomly select crossovers that are often not beneficial towards separating the problem into sub-problems without the aid of many repeated attempts over several generation iterations.

In practice, expert knowledge is sometimes used to place inputs, believed to be related, next to each other on the candidate chromosome string; increasing the chances that a single crossover operation will carry forward discovered sub-solutions. While this approach is reasonable, it assumes that the relationships for this domain will fall within previously observed behaviors.[89]

An advancement will be presented in Section 2.3 to the common use of genetic algorithms in problem decomposition. The presented method gains its efficiency by modeling these inter-variable relationships and leveraging this information when solving a problem.[41] To provide context for the later advancement in modeling inter-variable relationships, the following examples explain the three main methods currently used by genetic algorithms to perform the crossover operation. The implicit modeling of coupling behavior is suggested as one reason a GA requires a large number of function calls for successful problem decomposition and the later reconstruction of a more ideal composite solution.

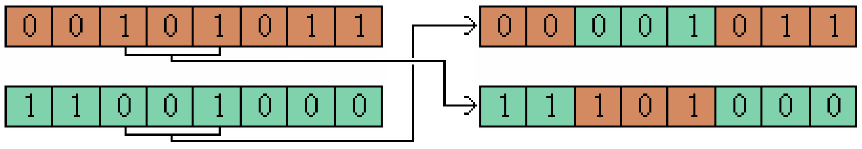


Figure 11: Two-point Crossover.

Two-point crossover increases the chances that candidates can be separated into their component sub-solutions during a single generation relative to using one-point crossover. This can help a discovered decomposition pass on during successive generations. Both indices are randomly selected and the points between them are swapped to generate two children, Figure 11.[23] Inputs from the start and end of the candidate string, that deal with the same sub-problem, have a chance of being transferred together as one unit to the next generation.

Though the representational power has increased for a single generation, the space of possible representations to randomly explore has also been increased. With no explicit means of determining where the crossover points should be, the method still requires a large number of iterations.[41]

Two-point crossover has a similar type of structural representational constraint as one-point crossover. With two-point crossover, there is a limit to the ability of the crossover operation to express the separation of interleaved sub-problems in a single generation, as in Figure 12. In Figure 12, the same colored inputs deal with the same sub-problem. Randomly, at least two sets of proper splits must be done for the sub-problem solution to be separated and passed on to a later candidate.

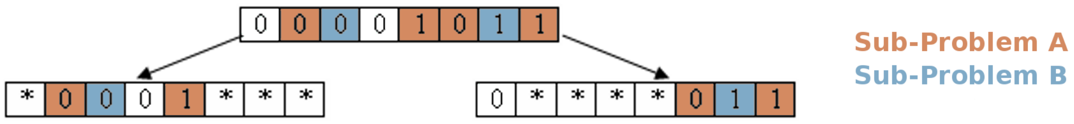


Figure 12: Two-point Crossover Performed on a Problem with Interleaved Sub-problems.

The uniform crossover operation further increases the single generation expressive ability of the crossover operation at the cost of a larger space of representations. Uniform crossover has every element within each parent as a starting and stopping index, inclusively, Figure 13. This means that each element has a chance of being traded between the parents; independent from the chance that any of the other elements were traded.[93]

The same colored elements in Figure 12 and Figure 14 represent separable sub-problems which are desirable to pass forward to the next generation, if possible. Though uniform crossover makes the transfer possible in one generation, the stochastic

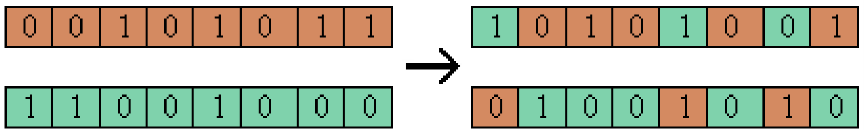


Figure 13: Uniform Crossover.

nature of the operation makes it unlikely in one generation.

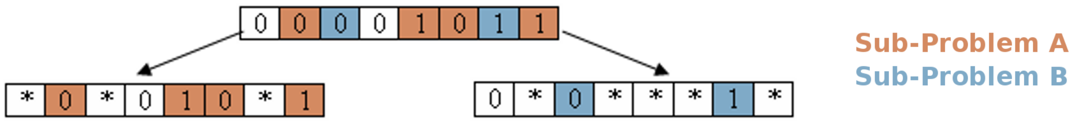


Figure 14: Uniform Crossover Performed on a Problem with Interleaved Sub-problems.

Uniform crossover is the most general in terms of its ability to separate sub-problems, as in Figure 14, but it also has the largest space of possible splits.[93] The biological inspiration for genetic algorithms is often quoted and is easy for an audience to grasp. Its easily understood operations, ease of implementation, and increases in computational power available have made it one of the main methods taught and used by engineers over multi-modal domains. Each of these three crossover operations is an unguided random selection of splits that assumes that well performing patterns, over many iterations, will stably remain within the population. Any increase in representational power for the expression of sub-problems needs to account for an increase in the space of all possible splits that must be searched by the random process; implying that a GA will often require many function calls to converge.

1.3.3.3 Evolving Neural Networks

Work by Khare[46] in 2006 sought to automatically decompose several problems by evolving neural networks (NN) to represent each sub-problem component. NN take a networked form inspired by the structure of neurons within the brain. They have great expressive ability for modeling non-linear functions and have been used widely within control systems and as a non-linear replacement for Response Surface Equations (RSE).[7] There are a large number of excellent introductions to the topic.[66]

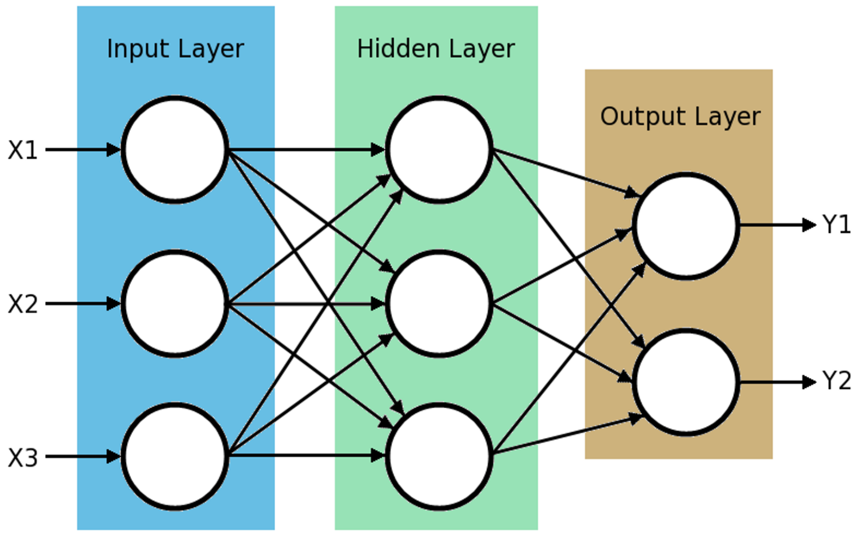


Figure 15: Neural Net with single hidden layer.

A hidden layer is composed of nodes that are not directly fed inputs and do not directly produce output values, Figure 15. Regarding the expressive power of neural networks, mathematical proofs have been completed that show that there exists a network with two hidden layers that can approximate any arbitrary function.[20] A network with a single hidden layer, such as the network in Figure 15, with a sufficient (possibly very large) number of hidden nodes can represent any continuous function and is capable of representing any boolean function.[20] One would use a neural network when there are a large number of varied examples for the behavior one wants to learn.

Training is performed by iterating through a training set, of inputs and expected outputs, multiple times. The inputs for each case are provided to the network and the resulting outputs are compared with expected results. Differences from the expected output values are propagated backward, through the network, to update link weights between nodes. The network regresses to model the behavior of the training set over several iterations. This set of data may contain errors as the training is robust to noise within the training set.[7] After a network is trained, the run-time evaluation of the regression is comparable to the speed of a large analytical expression.[37]

A neural network acts as a black box. It is capable, after training, of providing responses quickly but does not provide a human-readable line of reasoning that explains how the response was determined. Cases of medical diagnosis, for example, require an explained set of human-readable reasoning that is not available through a neural network. Provided a list of patient characteristics and symptoms a trained network could provide a diagnosis but doctors would also wish to know that the diagnosis was driven, for instance, by the level of iron in the patient's blood and his temperature. The network provides a diagnosis not an explanation for the diagnosis.

A set of validation data is required in addition to the training set to avoid the network over-fitting during training. Each link has an associated weight that signifies the importance that should be assigned to a signal being passed through that link. Training involves the updating of each weight to better predict the set of training data. When a network over-fits a set of data, both the information and noise from the training set is fit by the model. The validation set is used to cut off the NN training once the network no longer improves its prediction of data that was not used during training. If a validation data set is used to trim links from a network, an additional validation set should be used to serve as an unbiased judge of the networks future performance. These additional data sets increase the already large sample data requirements for the network.[66]

Khare's work on automatic problem decomposition [46] sought to use neural networks to regress the behavior of sub-problems automatically with the addition of genetic operators. Each module in the Module Population of Figure 16 is given a randomly chosen subset of the input variables and seeks to regress sub-section behavior by the examples present in the networks training set. This attempts to decompose those sub-problems that are solvable from global input information alone; as opposed to those that require the output from other modules. All of these models compete with each other to form the building blocks that are used by the larger modular neural networks, present in a separate System Population, Figure 16. Here the combining of the potential sub-problem modules is viewed as a problem in itself.[46]

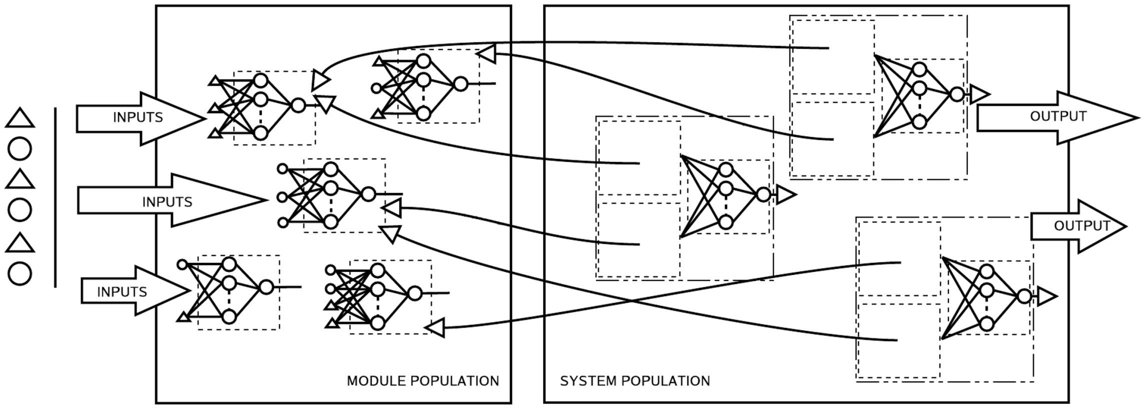


Figure 16: Two Populations Maintained to Generate Co-evolutionary Modules.[46]

The method randomly assembles sub-system modules that are then randomly assembled into composite system models that are then trained on training data. The presence of sub-system modules in high performing system models provide a fitness measure for the module population. The fitness measurement used to judge system performance is how well the formed networks are able to predict an available validation data set. One could also have other fitness metrics for the sub-system module population such as a measure of how often a sub-system module is used in the system population. This assumes that the average fitness of the system population will increase over time and be attributable to increasingly valuable component modules. The generation of new modules relies on genetically inspired operations to produce new modules that have a chance to be used by newly assembled systems in the system population. These would then be trained and ranked again, Figure 16.

The two population model shown in Figure 16 does not scale well with problem size. The training time required to evaluate the system performance increases with problem size to where it is inefficient to use genetic operations. The required time for training and validation means that the discovery of the correct number of sub-problems does not scale well for this method. This is shown by Khare[46], where the maximum number of sub-problems was set to three, to limit the possible combinations to train and search. It was recognized that an automatic decomposition method should be able to determine the number of sub-problems automatically. The genetically inspired searching method though was not able to scale well enough to set the upper limit at some very high number or remove the need for an upper limit.[46]

Modeling the coupling between inputs may aid this method by cutting down on the number of combinations for the networks to attempt. Large scale aerospace problems will need a method that scales well with the number of input variables for the automated discovery of sub-problems.

1.4 Research Objectives

The size of multidisciplinary problems that are considered to be feasible continues to expand with advances to hardware and decomposition methods. The problems tackled by engineers continue to evolve towards higher fidelity and wider domain exploration, always on the edge of current processing capability. To push ahead, toward solving currently infeasible unseparated problems, better methods and tools for problem decomposition must be developed. There is need for a new flexible heuristic to measure link importance so a discovered ranking can better inform dynamic decomposition efforts. Additionally, the static and dynamic decomposition methods used to rearrange a DSM into separable modules can be advanced by introducing a better global optimizer than the current state of the art in engineering.

1.4.1 Link Importance Heuristic

There is a need for flexible heuristics to evaluate the ranked importance of information links in a design analysis over the entire design space. Ranked links would aid in the dynamic clustering of internally coupled sub-problems. Any heuristics should be able to handle both continuous and discrete variables. In addition, a heuristic ideally should be efficient to calculate during the exploration of a design space. The link importance heuristic would ideally also be easily understood by informed engineers. For instance, it should be more expressive of the structure of the problem than the weights of a neural network. This work will provide a heuristic to compute the importance of variables in a design structure matrix using concepts from information theory. This heuristic will be applied to an entry study at Mars to judge the relative importance of analysis links used in the entry systems synthesis tool designed for the study.

1.4.2 Static Decomposition

Several methods in engineering have attempted to discover problem structure by rearranging analyses in a design structure matrix. The rearrangement of a DSM can reveal clustered groups of analyses. If the clusters are weakly linked these can be treated as separable sub-problems and shorten the required run time for a system evaluation of the DSM. Most of these methods rely on static heuristics or expert opinion that can be evaluated before running any of the analyses.

Several published utility functions have been shown for reordering a DSM and are normally applied to the problem by using a global optimizer. An advancement to the current practice, suggested by this doctoral work, replaces this global optimizer with another global method that explicitly models problem structure. By leveraging problem inter-dependencies, the proposed global method has the potential to reach useful decompositions with fewer utility function calls. The method, introduced in Section 2.3, is pulled from Computer Science and has its basis in the information theoretic construct of mutual information. This method has been further adapted for use in engineering. The use of automatic decision trees formalizes the optimizer performance comparison over a wide region of the static problem space.

1.4.3 Entry, Descent and Landing Design Synthesis

The Planetary Entry Systems Synthesis Tool (PESST) is a rapid conceptual design tool that was developed as part of this work for entry, descent, and landing systems. This framework has the capability to estimate the performance of an entry system using user-defined geometry, hypersonic aerodynamics, flight mechanics, thermal response, and mass estimation. Trade studies can be performed by parameter sweeps to gain a valuable understanding of the design space for conceptual studies. This framework is broadly applicable to conceptual studies of Entry, Descent and Landing (EDL) systems and will be used as a realistic test case for the evaluation of the proposed link importance metric.

Conceptual design decisions are better informed by the early application of physics-based tools to the design process. First order tools are often fast enough to be used during parametric studies while still providing information on many of the main effects and trades that can now be examined, far earlier in the design process, for entry missions. The application of this type of synthesis tool to early EDL design will be necessary to understand the system interactions of novel projected EDL mission concepts.[12] Details are provided in Chapter 5 for the methods used to implement each discipline module in this thesis.

1.5 Contributions of Work

This dissertation advances the state-of-the-art by making significant contributions across the domains of link importance heuristics, static decomposition, and entry, decent & landing design synthesis.

1.5.1 Link Importance Heuristic

1. **Introduction of a Link Importance Heuristic**: A concept from Information Theory, mutual information, is introduced as a useful link importance metric for ranking variables utilized in engineering problems. Mutual information is a novel metric for data passed through engineering design problems and works on both continuous and discrete data. The metric can measure both the linear and non-linear dependence between variables without the limitations of linear dependence measured through covariance. Mutual information is able to handle values that do not have derivative information unlike other metrics that require it. A graphical method, force-based clustering, is modified to discover related sub-graph structure as a function of DSM structure and link importance ranking. In this work, force-based clustering will utilize mutual information to determine link importance.

1.5.2 Static Decomposition

2. **Introduction of a Global Optimizer for Static Decomposition**: Genetic Algorithms are a global method typically utilized to apply utility functions for static decomposition. An advancement to the current practice replaces this global optimizer with another global method that explicitly models problem structure. By leveraging problem inter-dependencies, the proposed global method has the potential to reach useful decompositions with fewer utility function calls. The inclusion of Parzen-window density estimation, performed in

this work, allows for the method to treat continuous variables. This global optimization method can now be used to operate over many aerospace multimodal domains containing discrete and continuous variables.

3. **Generalizing Static Results with Automatic Decision Trees**: The vast majority of work on decomposition methods in engineering focus on results for single problem instances, without trying to develop descriptive metrics for quantifying performance over a larger region of the problem space. By leveraging automatic decision tree generation methods from Machine Learning and a randomly created problem set, a decision tree describing which method to apply for the static decomposition of the problems examined is created.

1.5.3 Entry, Descent and Landing Design Synthesis

4. **Creation of a Design Synthesis Tool for Entry Design**: The robustness of the link heuristic will be tested by application to realistic entry conceptual design problems. The PESST framework has been developed as part of this work and will serve to demonstrate link importance calculations on a tool with realistic discipline behavior. PESST fills an important place in the field of entry design synthesis by allowing a designer the chance to quickly gain an understanding of a design space using first-order physical models.

1.6 Summary

Though the scale of feasible multidisciplinary problems has been increasing over the past twenty years, improvements in both processing power and decomposition methods will be required to push the solution of even larger problems into feasibility. Further improvement to the currently utilized decomposition methods in engineering would better advance the capabilities of algorithms to form lower-order sub-problems. Several of the most commonly used methods in engineering were examined and areas available for improvement are described in Chapter 2. Chapter 3 examines the performance of a newly introduced variable importance metric on the design of a conceptual entry mission and the use of force-based clustering to decompose a low-thrust entry problem. Chapter 4 provides the framework and experimental results on a new method suggested for static decomposition and its comparison against current practice. Chapter 5 examines the discipline analysis modules produced for the PESST framework and Chapter 6 provides a summary of the results from this doctoral work.

CHAPTER II

METHODOLOGY

2.1 Mutual Information

Mutual information is suggested as a useful metric for judging the importance of data links in a design structure matrix. Mutual information comes as a concept from information theory based on the foundational work of Claude Shannon.[87] It estimates the amount of shared uncertainty or dependence present between two random variables. This metric can be easily computed for each of the links in a design structure matrix. Though several of the techniques used to estimate mutual information could be adapted to estimate partial derivatives; mutual information measures the global general dependence between two random variables and can be used with both continuous and discrete data; this includes naturally discrete data (EngineA, EngineB, etc) where a derivative is unavailable.

Correlation is commonly used in engineering to measure dependence between random variables. Correlation only measures the linear dependence between two variables while mutual information measures general dependence.[57] For example, a large random data set from two variables x and $y = sin(x)$ would have a correlation converging to zero. A data sample taken from a small region would show a strong positive, weak positive, neutral, weak negative, or a strong negative correlation depending on the region sampled; one is using a linear tool to estimate non-linear dependence. Both linear and non-linear dependence can be analyzed using mutual information.

A partial derivative, as another tool to show dependence at a single point, is not able to describe the behavior of variables that do not possess meaningful derivatives.

Mutual information does not have this limitation. Partials also provide local information while mutual information works with the global behavior of the variables. To better understand the concept of mutual information, the concept of entropy will be examined briefly.

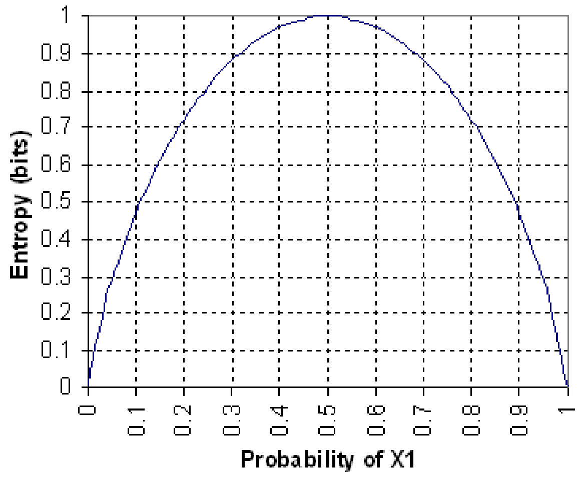


Figure 17: Entropy in the Case of Two Possibilities with Probabilities p and (1-p).

Entropy is a concept borrowed from chemistry and is considered here as a measure of the uncertainty associated with knowing the value of a random variable. In the graph shown in Figure 17, the entropy is highest when there is an equal chance of the random variable having the value of 0 or 1. When there is a 100 percent chance of the random variable having a value of 1 there is no entropy, or uncertainty, associated with the variable. This is also true when the variable has a 100 percent chance of being equal to 0. It makes intuitive sense that the state containing the most uncertainty is one where there is an equal chance of any number of values occurring. The formula to calculate entropy is shown in Equation 2 for continuous variables and in Equation 3 for discrete variables.

$$H(X) = -\int_X p(x)\log_2(p(x)) \tag{2}$$

$$H(X) = -\sum_{x \in X} p(x) \log_2(p(x)) \tag{3}$$

The reader should note that only the marginal probability distribution $p(x)$ is required to compute the entropy of a random variable. Log base two is used in information theory as one normally speaks in terms of a binary encoding of information transmitted over a communication channel. By presenting entropy this way, it expresses the number of bits required to remove the uncertainty present in the random variable. By using Equations 2 and 3, one can see that two bits are required to remove the uncertainty present in a random variable that possesses four equally likely options, see Equation 4.

$$\begin{aligned} H(X) &= -\sum_{x \in X} p(x) \log_2(p(x)) \\ &= -[0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25] \\ &= -[\log_2 0.25] \\ &= -[-2] = 2 \text{ bits minimum to describe option selected} \end{aligned} \tag{4}$$

More intuitively, having four options in binary requires at least two bits to name them, {00, 01, 10, 11}, if all the options are equally likely. If one option is more likely than the others, there would be less uncertainty regarding the answer and therefore fewer bits on average would be required to name each option. An encoding for the case where probabilities for the options fall as (0.5, 0.2, 0.2, 0.1) would have an average bit length of 1.761 bits as a theoretical minimum. One example encoding for this case would be 0, 10, 110, and 111 (i.e. 0 is used 50% of the time, 10 used 20%, 110 used 20%, and 111 used 10% of the time). For this example encoding, (1 bit $\times 0.5$)+(2 bits $\times 0.2$)+(3 bits $\times 0.2$)+(3 bits $\times 0.1$) $= 1.8$ average bits are used. This is close to 1.761 bits which was what was calculated as the theoretical minimum required to describe the data, see Equation 5.

$$\begin{aligned}
H(X) &= -\sum_{x\in X} p(x)\log_2(p(x)) \\
&= -[0.5\log_2 0.5 + 0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.1\log_2 0.1] \\
&= -[-0.5 - 0.4644 - 0.4644 - 0.3322] \\
&= -[-1.7610] = 1.761 \text{ bits minimum to describe option selected} \qquad (5)
\end{aligned}$$

The entropy for a random variable can also be viewed as the information present in that random variable. The number of bits necessary to describe the information contained in the random variable. If a variable has a 100 percent chance of being equal to one, there is no new information that would be gained by knowing the value of that variable. This is a way of quantifying the information that would remove the uncertainty of a random variable.

Mutual information is the amount of entropy, aka information, that two random variables share. If two random variables are independent learning the value of one variable, gaining its information, will not reduce the entropy still possessed by the other random variable.

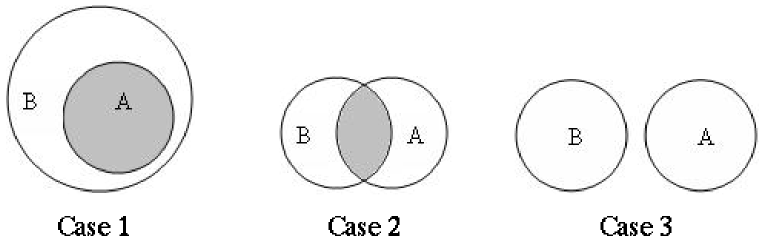


Figure 18: Three Cases of Mutual Information Between Two Variables.

The mutual information between the variables A and B, I(A,B), is the shaded area shown in each of the three cases in figure 18. This is how much information one random variable tells us about the other. If someone knew B in Case 1, all the uncertainty in A would be explained. If there is an amount of shared uncertainty, this is the mutual information. When the value of one random variable becomes known, the uncertainty shared with the other random variable is made certain, lowering the entropy in the remaining random variable. The amount of information that the remaining variable can provide has been reduced.

$$H(X,Y) = -\int_X \int_Y p(x,y)\log_2(p(x,y)) \tag{6}$$

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2(p(x,y)) \tag{7}$$

Calculating the amount of mutual information requires the calculation of the joint entropy which is the total sum of unique information provided by two random variables. In Venn diagrams, such as Figure 18, it would be the total area represented by both circles without double counting any overlap. This can be computed by Equation 6 for continuous variables or equation 7 for discrete variables.

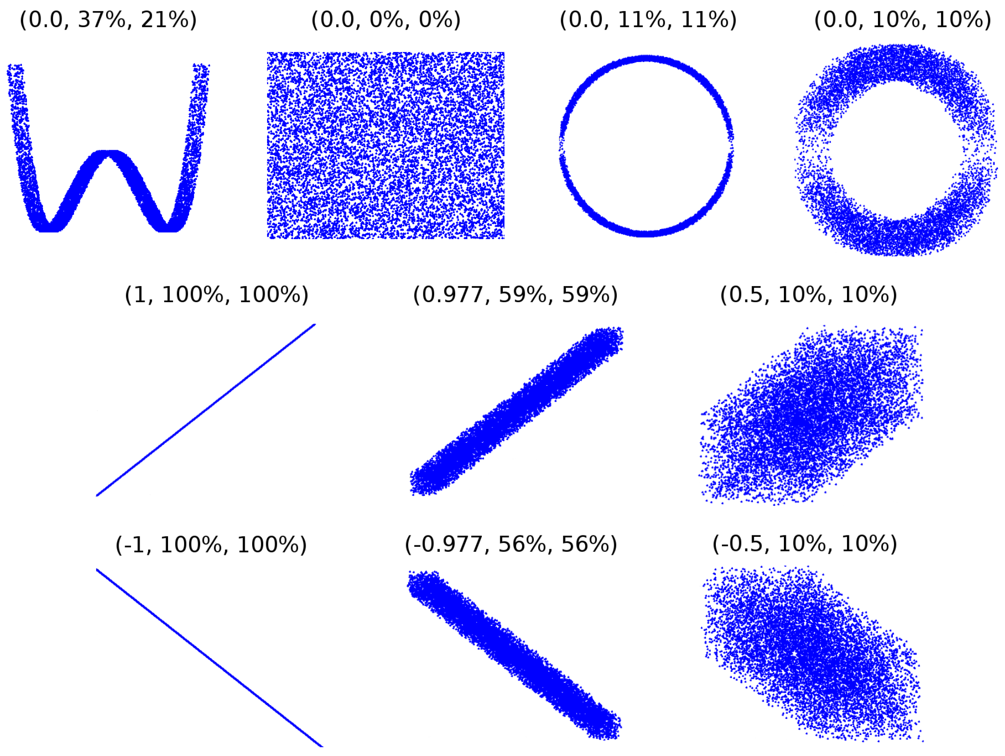
$$I(A,B) = H(A) + H(B) - H(A,B) \tag{8}$$

Equation 8 shows that, when calculating the mutual information, one can pictorially consider it as the area of overlapped entropy between two random variables. The calculation of the mutual information requires knowledge of the marginal probability distributions for the random variables, to calculate $H(A)$, and the joint probability distribution between variables, to calculate $H(A,B)$.

The probability distributions used to calculate mutual information can be estimated from a finite set of sampled data allowing for the use of mutual information for continuous and discrete data. As either continuous or discrete distributions can be used, the discrete data does not have to come from a discretized continuous range. A set of values for EngineA, EngineB, etc. works just as well as the set of real numbers.

Figure 19 shows ten examples of joint distributions between the random variables X and Y. The format of the label for each example is provided in the Figure. The

Pearson correlation is calculated for each example. The value computed for mutual information is the bits of shared entropy between the random variables X and Y. The mutual information is normalized by the total entropy to show the percentage of the random variable entropy shared by the other variable.



$$\left(Correlation, \frac{I(X,Y)}{H(X)}, \frac{I(X,Y)}{H(Y)}\right)$$

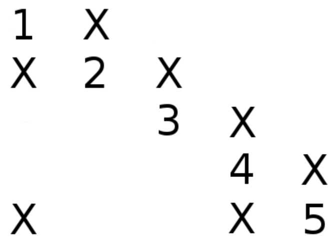
Figure 19: Examples of Correlation and Mutual Information on Joint Distributions.

The correlation provides directional information that is not present in the value for mutual information but mutual information is able to provide a measure of dependence for non-linear relationships where correlation could be equal to zero. When there is no noise between the relationship of X and Y, the mutual information is 100%. All of the uncertainty of X is explained by a value for Y. Random noise lowers the shared uncertainty between the two variables though it may not always lower the magnitude of the correlation. For the case where X and Y have no relationship, both correlation and mutual information agree that there is no dependence relationship.

Mutual information does not deal with the local behavior of the variable at one point.[57] The value for mutual information is measured over the range of the probability distribution estimated by sampling from the space of solutions. This global view provides a better general metric when evaluating decisions for the potential decomposition of larger problems. This type of global information should provide a more robust decomposition than if one were to have only used locally accurate information.

2.1.1 Mutual Information as a Link Rank Metric

A metric to evaluate the importance of information links between disciplines is valuable as one can gain insight into the relationships driving the problem. This data would be useful when planning which groups should work closely together on a project. As the structure of a problem is better understood, the problem can often be made more efficient or a provided solution made more robust. For example, no link information is known in Figure 20. An engineer seeking separable sub-problems may wonder what arrangement would lead to the best set in interior connections with the weakest external connections to other groups. The engineer can either assume that the links are equally important or seek to estimate the importance that should be assigned to each link. The more information that is available during a decomposition process, the more informed a sub-problem separation can become.

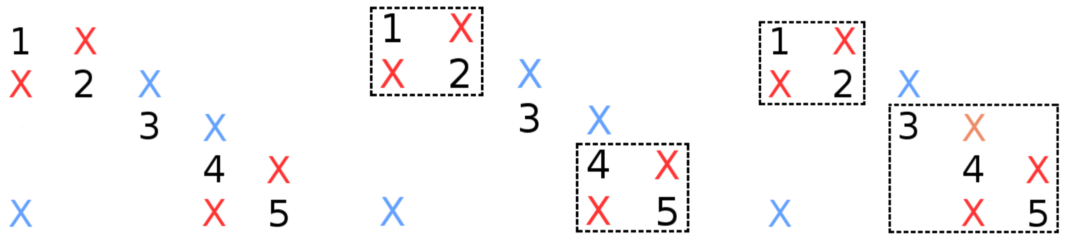


$$\begin{array}{ccccc} 1 & X & & & \\ X & 2 & X & & \\ & & 3 & X & \\ & & & 4 & X \\ X & & & X & 5 \end{array}$$

Figure 20: DSM Example with No Link Importance Information.

Adding link knowledge is much like adding color to the DSM in Figure 20. Two closely coupled problems are plainly seen with strong interior links in Figure 21(b).

These clusters would have been considered when the links were considered as being equally weighted but an equal weighting would not have provided guidance on where analysis 3 should be placed. Link information and structure guides sub-groupings. Updated link information can be used to update the groupings utilized for the problem, Figure 21(c). Ideally most of the information required to solve a useful piece of the problem should be found within a grouping with a minimum of required data passed into the grouping. This leads to separable sub-problems with the potential for concurrent evaluation. The decomposition provides insight into the mechanics of the problem and provides the potential to lower the time involved in finding a system solution.



(a) Links Weighted (b) Two Sub-problems Marked (c) Updated Link Implies new Sub-problem Structure

Figure 21: Importance of Link Ranking to Forming Sub-problem Clusters.

Applying this ranking metric to the problem of decomposition, a workable metric could be used to select better quality sub-problems. A high quality sub-problem representation would display groups of strongly connected components, with highly ranked connections, as opposed to connections that are not as valuable for the requested analyses. A metric could also be used to select links that could be temporarily removed to create a better performing decomposition; allowing the option to temporarily remove less important links from an analysis.

The current link metrics examined in Chapter 1 have several drawbacks; for instance, treating all links as having the same importance fails to capture useful information about the problem that could be used to more effectively decompose the original problem. Experts may not always be available or their intuition may be misleading on a novel problem. The number of variables in the link does not necessarily relate to the importance of these variables. All of the passed variables in a link could be unimportant. A metric is needed that can adapt to the problem at hand.

The use of partial derivative information is dynamic to the problem at hand but is only a good estimate for the local area around the point where the partial derivative is taken. It provides local importance information and can only be calculated where a derivative exists; continuous or discretized continuous variables. Naturally discrete variables that are not discretized from a continuous source (EngineA, EngineB, etc) can not have their partial taken. Ideally, a workable metric would be able to also handle these naturally discrete variables. Inexpensive analytical derivatives are also not often available on realistic engineering problems.

2.2 Force-Based Clustering

Future work on dynamic decomposition is supported by the introduction of force-based clustering. The graphical clustering method discovers sub-problem structure based on linkage connections and their ranked importance. A future dynamic method could use force-based clustering to determine a temporary configuration of clusters before re-ranking link importance based on computed analysis results. In this work, mutual information will be used as a calculated importance metric for each of the links in a design structure matrix. This does not imply that mutual information is the only metric that could be applied with force-based clustering. The algorithm is independent of the metric used to rank the importance of links and future metric developments or past expert-based ranking could also be used as input.

A force-based graphing approach rephrases the problem to something similar to an n-body problem. Analysis vertices that are closely coupled together by links will be drawn into clusters while weaker linked vertices will drift farther away from the cluster.[27]

Force-based clustering is demonstrated here by returning to the large DSM problem that was earlier shown in both a randomized and ordered form in Figure 22. There is a desire to obtain informative groupings that can be used for the analyses of the problem and to aid in understanding the processes at work in the problem. The ordered DSM shown in Figure 22(b) tells much more about the structure of the problem than the unordered DSM in Figure 22(a).

Force-based clustering works by providing the analysis nodes a repulsive force towards other nodes. Every link is given an equal attractive force that pulls its end nodes towards each other. This method forms groupings based on the interconnections between the analyses.[27] The unmodified method could be used as another method of decomposing DSMs before executing any of the analyses.

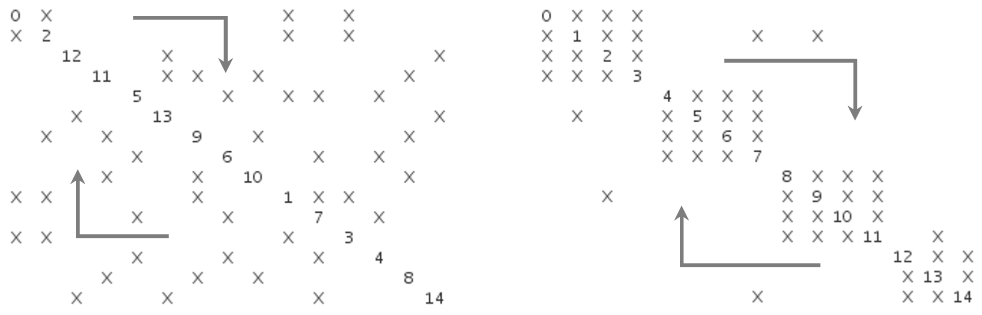

(a) Randomly Ordered DSM (b) Decomposed DSM

Figure 22: An Ordered and Unordered DSM Example

Force-based clustering is a local method that may create an arrangement at a local minima. The initial locations used for the nodes, before repulsive forces are applied, will change their final locations in the graph and may change the clarity of the resulting clusters. The search method though could be leveraged in static decomposition, when the link rankings are considered as equally important.

Standard force-based methods can be modified slightly to allow for static decomposition with equally ranked links or for use in dynamic decomposition. The link attractive strength has been modified and made proportional to the importance of the link, potentially forming different groupings than when using equally ranked links.

When the avoidance of local minima is worth the likely increase in computational cost of a global method, MIMIC and genetic algorithms provide a means for reorganizing a DSM. For dynamic decomposition, the assumption is made that a quick decomposition performed by local force-clustering methods will be sufficient to update how the problem should be separated between iterations. In dynamic decomposition the sub-groupings should be flexible, new information regarding the ranking of a link may change the sub-problem an analysis is assigned to. The sub-groupings for a problem that are discovered midway towards a solution may not be the final groupings used when more information is known regarding the problem. The ability of force-based methods to provide an adequate decomposition based on the current data can be provided quickly to the scheduler running the execution of the problem. If a global solution is needed for the organization problem, the MIMIC algorithm or genetic algorithms are available to future researchers of dynamic methods.

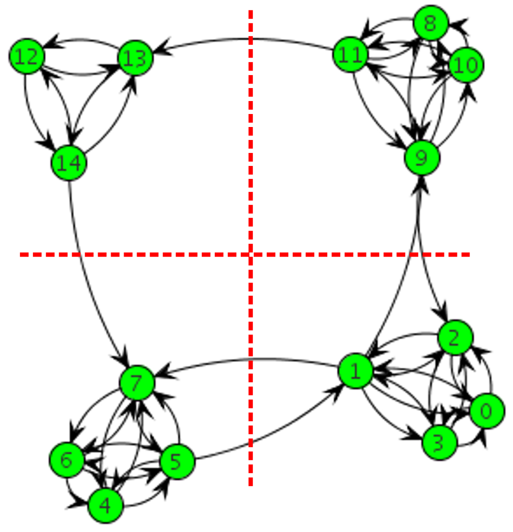


Figure 23: Using Force-Based Clustering to Discover Clusters and Cluster Interfaces

Force-based clustering was used on the randomized DSM problem in Figure 22(a) and was able to discover a useful decomposition shown in Figure 23 that is equivalent to the ordered DSM in Figure 22(b). For this example, the links were ranked as equally important and non-varying in time, making this equivalent to solving for the static reorganization of the DSM problem.

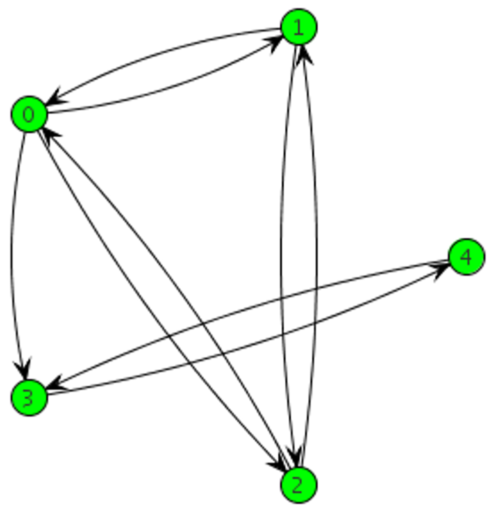
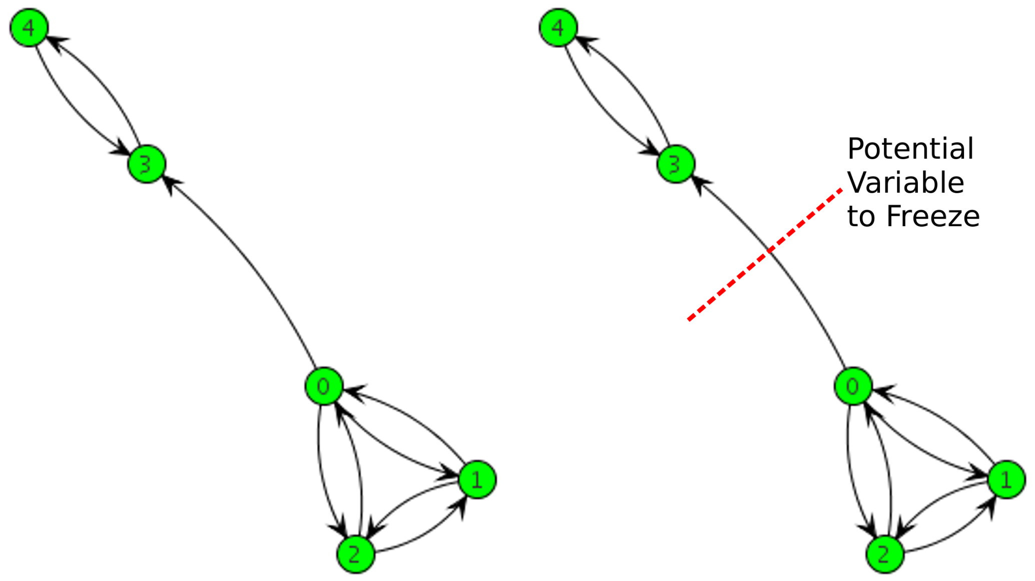


Figure 24: Sample Connected Graph

Using force-based clustering on the problem in Figure 24 creates the graph shown in Figure 25(a). Here also every edge is considered as equally attracting. The modification that makes this method useful for future work in dynamic decomposition is to adjust the attractive force for edges so that they are a function of the edge importance rank. The leads to potentially different clustering configurations as more becomes known regarding the problem during execution.

The sub-graphs that have been generated in Figure 25(b) would show the sub-problems implied by the link ranking. Data from the low-importance interfaces, between the sub-graphs, could be minimally updated while the sub-graphs converged on sub-solutions. The graphs would then be recombined to converge onto a solution in the original problem space. At that point, each sub-problem found would have been initialized by the sub-domain search and have a potential to dramatically cut into the combinatorial space of solutions.



(a) Example Force Clustered Graph (b) Possible Interface between Clusters

Figure 25: Example of Force-Based Clustering

2.2.1 Determination of Sub-Problem Clusters

When the attractive force of edges are proportional to the mutual information weight assigned to the edge, clusters will form between analyses that share a large number of high quality links. Analyses with few low importance links or those with one mid-importance link will be driven further from the discovered clusters. In Figure 25(b) and Figure 23 the first links to consider for removal are the longest links in the graph while the links that are desirable to keep, for clustering, are the shortest links.

If the process of graph link removal is to be automated, a cluster quality metric is required to judge when removing links is no longer increasing the quality of the remaining clusters. Cluster quality metrics remain an area of active research but results to date generally demonstrate the benefit of forming closely coupled groups of analyses with few interconnections between analyses.[83] Though the discovery of which clusters to form is normally computationally difficult, the evaluation of a given cluster can often be done in polynomial time.[83]

A link density metric can be used to evaluate sub-graphs created by link removal. If the sub-graph created by the removal of a link is densely-connected, it is considered as a high quality cluster. As an example of the difficulty of the problem, the discovery of a way to partition a graph into n equally-sized sub-graphs where the number of links to cut has been minimized has been found to be NP-hard.[29]

2.3 Static Decomposition

Pre-execution or static decomposition is a very common set of methods used to automatically separate engineering problems into sub-problems.[95, 99] Methods are grouped into this category by their aim to rearrange and separate the problem components before executing any of these components. The methods either assume that all links are equally valuable[91] or that they have values based from metrics that can be computed without evaluating the connected analysis tools.[78, 79] Expert surveys, the number of variables handled by the link, and the location of the link in a DSM are all metrics that have been used to rank link importance. A high performing arrangement for the problem is normally then found using a global optimizer with a utility function that incorporates these link ranking metrics. In engineering, a genetic algorithm has most often been utilized for this global search.[1]

A strong improvement to this current state of the art in aerospace engineering would be to improve the global optimization method utilized to explore the space of potential problem arrangements when decomposing a problem statically. This optimization method is extracted from recent work in Computer Science and displays several advantages to the use of genetic algorithms when exploring spaces with a structural relationship between the input variables. The problem of arranging disciplines in a design structure matrix has a great deal of structure that can be leveraged to converge towards better performing solutions with fewer function calls than typically taken by a GA. Mutual Information Maximizing Input Clustering (MIMIC) is described below and is potentially useful for this and many multimodal domains in aerospace. Originally demonstrated for problems with discrete input variables, an addition to the treatment of the variables detailed in this work will enable the use of MIMIC with continuous input variables.

2.3.1 MIMIC a New Optimizer for Static Decomposition

Mutual Information Maximizing Input Clustering (MIMIC) was designed by taking a powerful concept from GAs, crossover, and refining it to compute where larger problems may best be separated into sub-problems. It is not another type of genetic algorithm. MIMIC is able to more efficiently converge over a large design space, containing structure that can be leveraged, by explicitly modeling the structural interactions between the input values.[8] It uses prior solutions to build a model of the solution space that focuses on areas of the space that are likely to contain high performing candidate solutions. It builds a modeled distribution over the solution space by using statistics from prior samples of the true distribution over all solutions.[5, 8]

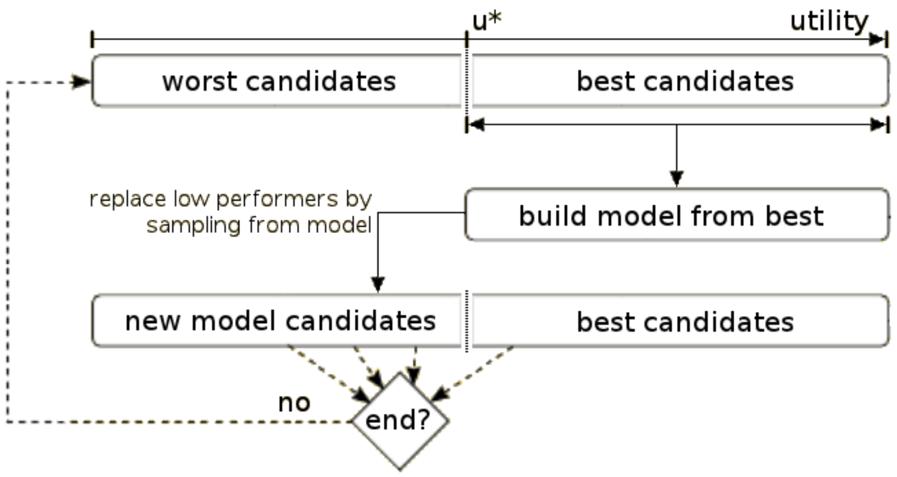


Figure 26: Overview of MIMIC Iteration Flow.

The model of the high performing candidates is used to replace poor performing solutions from the candidate list, see Figure 26. How the model for this high performing region is formed and sampled from is examined in greater detail below. The essential idea is to closely model the regions described by a set of high performing candidates so that future candidates can be sampled from these regions. This will improve the average performance of the candidate list over time until the method registers convergence.

2.3.1.1 Building the Model for a Sampled Region

Equation 9 describes the exact joint distribution between a set of random variables. By approximating this function through pair-wise relationships, information about the relationships between the variables can be exploited to search the domain space. The dependency tree version of MIMIC[5] uses pair-wise conditional probabilities to create an approximation to the true distribution.

$$p(\bar{X})_{true} = p(X_1|X_2...X_N)p(X_2|X_3...X_N)...p(X_{N-1}|X_N)p(X_N) \tag{9}$$

This approximation could be made to use ternary conditional probabilities [ie $p(x \mid Y = y, Z = z)$] for a more accurate match to the exact joint distribution but would require far more data to accurately determine the ternary interactions.[8] Pair-wise conditional probabilities [ie $p(x \mid Y = y)$] are used to approximate the true joint distribution in Equation 9 as they are easier to determine from a smaller set of data and was found to be a sufficient approximation to the joint distribution for the problems examined in this work.

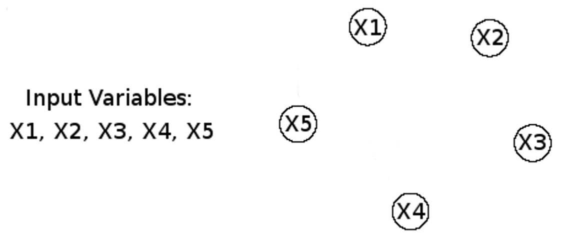


Figure 27: Specifying Nodes for Each input Variable.

To show how the statistical data is used to create a distribution model, each variable is first assigned a node in the graph that will later serve as the approximated model, Figure 27. The mutual information between each pair of variables i and j is computed; this only requires knowledge of $p(x_i)$ and $p(x_i \textit{ given } x_j)$. The mutual information can be calculated from this information and used to weight the links between the variables. A fully connected graph with computed weights is shown in Figure 28 for an example problem. Each edge value is the mutual information between the two variables; this can take any positive value inclusive of zero. Variables with four bits worth of shared information would have an edge value of four, for instance.

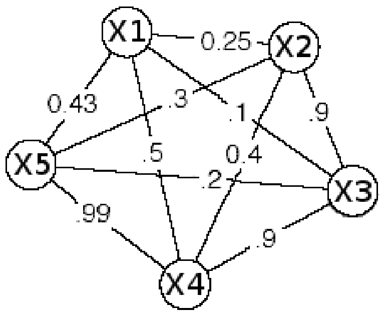


Figure 28: Fully Connected Graph Showing Mutual Information Between Each Node.

The Kullback-Leibler (KL) distance between the approximated distribution and the exact or true distribution is a measure of how similar the two distributions are.[51] By definition, the model with pair-wise statistics that minimizes the KL distance between it and the true distribution would be the best possible pair-wise approximation for the true distribution.[18] Chow[18] and Baluja[5] have found that the graph that minimizes the KL distance in a pair-wise model is a maximum spanning tree where the edges are weighted by mutual information.

To form this maximum spanning tree, Figure 29, one simply keeps the highest weighted links between the nodes to form an acyclic tree containing every node. One can use Prim's algorithm to automatically discover the tree structure, changing it to find the maximum spanning tree instead of the traditional minimum spanning

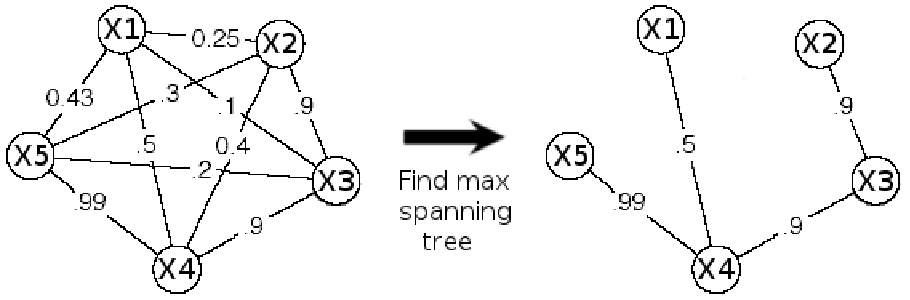


Figure 29: Finding the Maximum Spanning Tree.

tree, Figure 29. This is often as simple as changing the less-than symbol used in an implementation of Prim's algorithm to a greater-than symbol. Or, equivalently, the weights can be negated and the minimum spanning tree found.

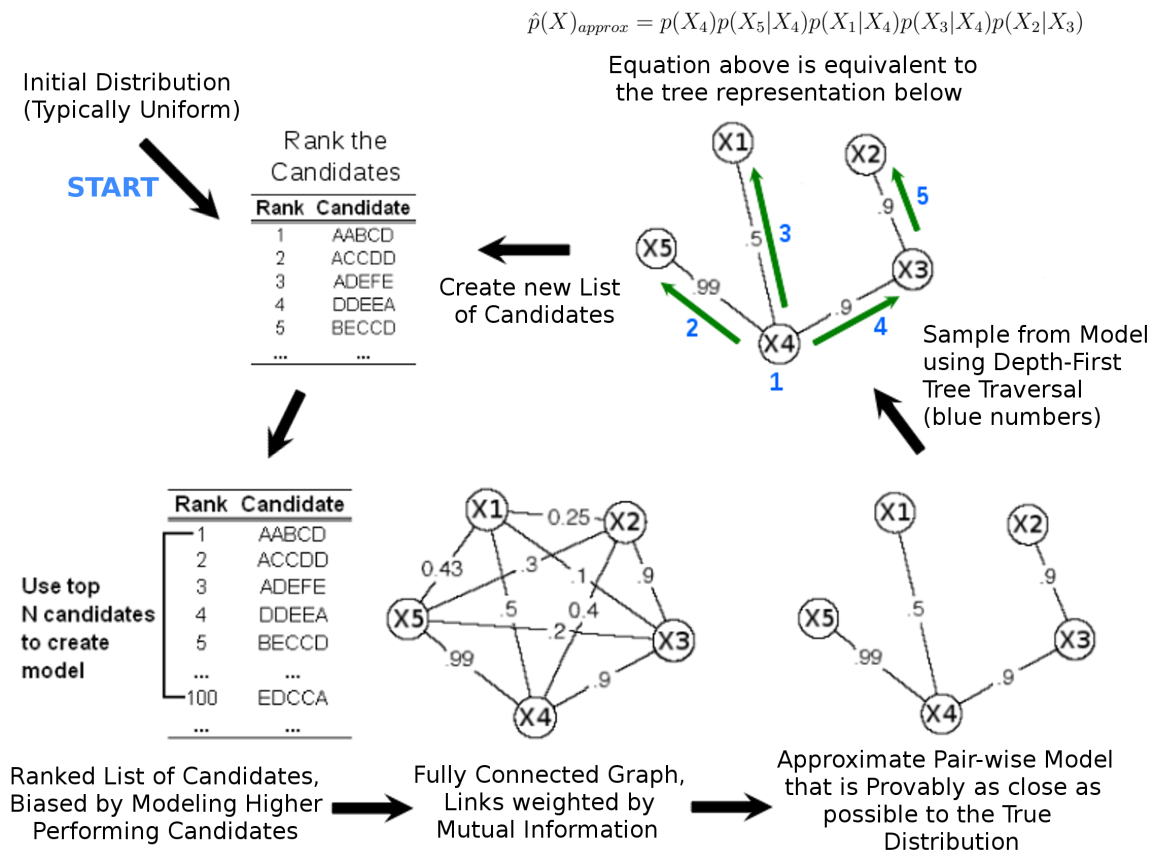
$$p(\bar{X})_{approx} = p(X_4)p(X_5|X_4)p(X_1|X_4)p(X_3|X_4)p(X_2|X_3) \tag{10}$$

This tree model for the distribution is equivalent to the best pair-wise approximation for the true distribution and requires only marginal and conditional probabilities. The tree shown in Figure 29 is equivalent to the distribution described by Equation 10.

2.3.1.2 Sampling from the Created Model

Sampling from the tree model is straightforward. The user selects any node and uses the probabilities for each option available to that node to select its value. Based on that value, a depth first transit (follow the path taken by a depth first search) is performed through the tree. At each node a value for that variable is determined based on the conditional probability of its value and the value taken by its parent on the tree. The estimate in Equation 10 is equivalent to the tree shown in Figure 29; The value for X_4 is discovered first, then the value for the child X_5 based off of $p(X_5|X_4)$. If X_5 had children their value, for the candidate input vector, would be computed next.

Now that the solution can be approximated by a joint distribution, the model can



$\hat{p}(X)_{approx} = p(X_4)p(X_5|X_4)p(X_1|X_4)p(X_3|X_4)p(X_2|X_3)$
Equation above is equivalent to the tree representation below
Initial Distribution (Typically Uniform)
START
Rank the Candidates
Rank Candidate
1 AABCD
2 ACCDD
3 ADEFE
4 DDEEA
5 BECCD
Create new List of Candidates
Sample from Model using Depth-First Tree Traversal (blue numbers)
Use top N candidates to create model
100 EDCCA
Ranked List of Candidates, Biased by Modeling Higher Performing Candidates
Fully Connected Graph, Links weighted by Mutual Information
Approximate Pair-wise Model that is Provably as close as possible to the True Distribution
X1
X2
X3
X4
X5

Figure 30: Flow of execution for the MIMIC algorithm.

be used to sample from higher performing areas. This model is biased towards the higher performing areas of the solution space by choosing to only create the model from the top N percent of the data. In MIMIC, N is typically set to 50 percent. The comparison with GAs in Chapter 4 is performed over a range of values for N. Sampling from the model is continued, to replace the lower performing candidates. The best N percent from the new candidate list is used to create the model for the next iteration. This biased model for the domain is focused on the better performing areas of the solution space. See Figure 30 for the pictorial flow of the MIMIC algorithm.

2.3.1.3 Analytic Demonstration for Leveraging Problem Structure

The four peaks problem is composed of two sub-problems whose solutions can conflict with each other. When the searching method is able to balance the needs of both solutions, a bonus is provided to the solution utility. This problem serves as an example to illustrate that MIMIC provides an improvement to the current practice which utilizes genetic algorithms when the problem displays structure that can be leveraged.

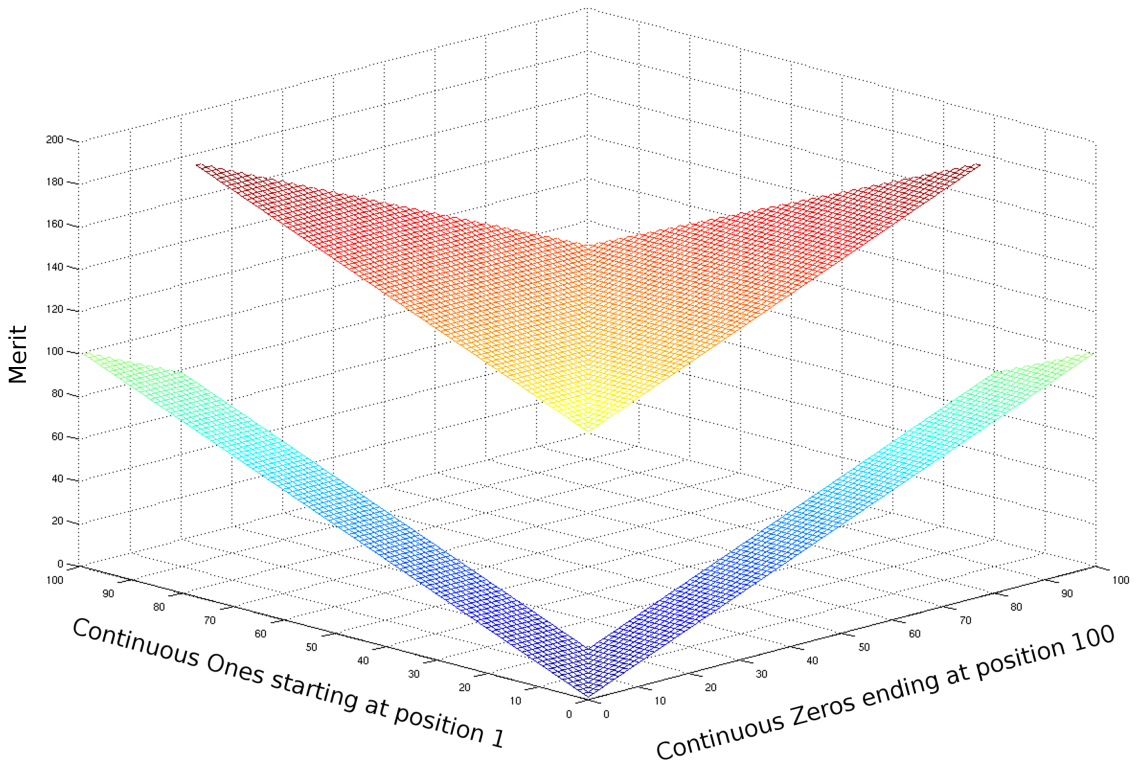


Figure 31: Four Peaks (Side View) for an input $\bar{X}$ of size 100, T=10.

The four peaks problem was used by Baluja to describe the behavior of PBIL[4] and later by DeBonet et al. for MIMIC[8]. This demonstration includes a confidence bound around the mean performance for MIMIC and Genetic Algorithms which was not included in earlier work.

The problem has two local minima, a string of all 1's or all 0's, and two global minima; either N-(T+1) leading 1's, or trailing 0's, with the remaining T+1 of the opposite value, see Figure 31. N is the number of values in the candidate string. T is a specified input that affects the size of the discontinuous raised region shown in Figure 31. The utility of a solution is primarily judged by the longest list of '1' values from the start of the array or the list of '0' values from the end of the array, depending on whichever is longer. When the lists are balanced so that they are both above the value of T, there is a reward to the utility function. Leading and trailing values are explained by example in Equation 11.

$$Example\ Candidate: \left[\underbrace{11111}_{\text{head=5}}010110111\underbrace{000000}_{\text{tail=6}}\right] \tag{11}$$

A mathematical description for the four peaks problem is shown in Equations 12. The value T is the min number of values in a row from both the head and tail that must be obtained before the utility reward is provided. For the example in Equation 11 which has $N = 20$ values, if $T = 6$ then the utility of the example is 6. If the problem specified that $T = 3$, then the solutions for the head and tail are both sufficiently long for the reward providing a utility of $6 + N = 6 + 20 = 26$.

$$\bar{f}(\bar{X},T) = max[tail(0,\bar{X}), head(1,\bar{X})] + reward(\bar{X},T) \tag{12a}$$

$$tail(0,\bar{X}) \text{ equals the number of trailing 0s in } \bar{X} \tag{12b}$$

$$head(1,\bar{X}) \text{ equals the number of leading 1s in } \bar{X} \tag{12c}$$

$$reward(\bar{X},T) = \begin{cases} N \text{ if } tail(0,\bar{X}) > T \text{ and } head(1,\bar{X}) > T \\ \text{0 otherwise} \end{cases} \tag{12d}$$

This example problem allows scaling of the problem size, by changing the size of N, and modification of the size of the raised platform in the solution space, by changing T. For this comparison, T is kept at 10 percent of the size of the input vector N. The solution space formed by this is shown in Figure 32 for a 100 input vector N. As the value of each input depends on the values of several other inputs, methods that explicitly model the inter-dependencies between inputs should be able to leverage this information to converge with fewer function calls.

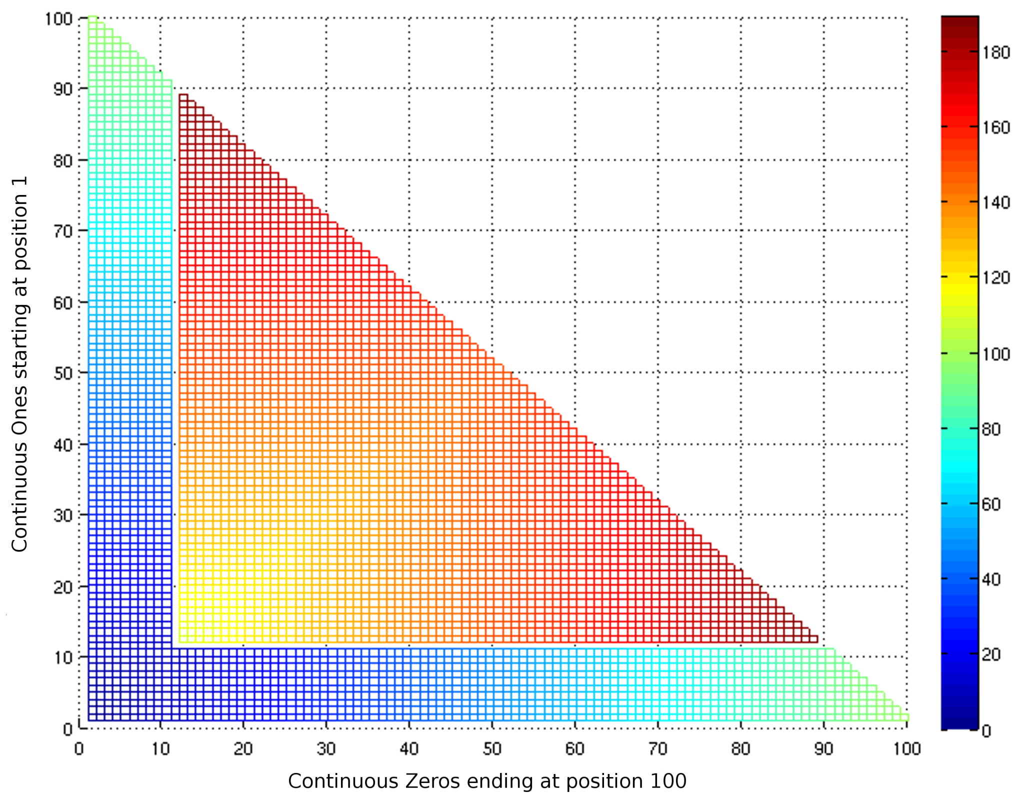


Figure 32: Four Peaks (Top View) for an input $\bar{X}$ of size 100, T=10.

A mature third-party implementation of genetic algorithms[58] is compared against an author created implementation of MIMIC with dependency trees. For the genetic algorithm, with a population of 100, tournament selection is utilized with the probability of crossover at 100 percent. Elitism is used to retain the best performing candidate from the last generation; mutation is kept at 5 percent. An initial search of GA settings was used to discover settings that consistently performed well across the range of inputs.

A full factorial search of GA settings was not performed so it can not be stated definitively that this is the best possible performance for Genetic Algorithms on this problem. This work also exemplifies the challenge of finding well performing GA settings.

Figure 33 shows the challenge two crossover operators (single and two-point crossover)

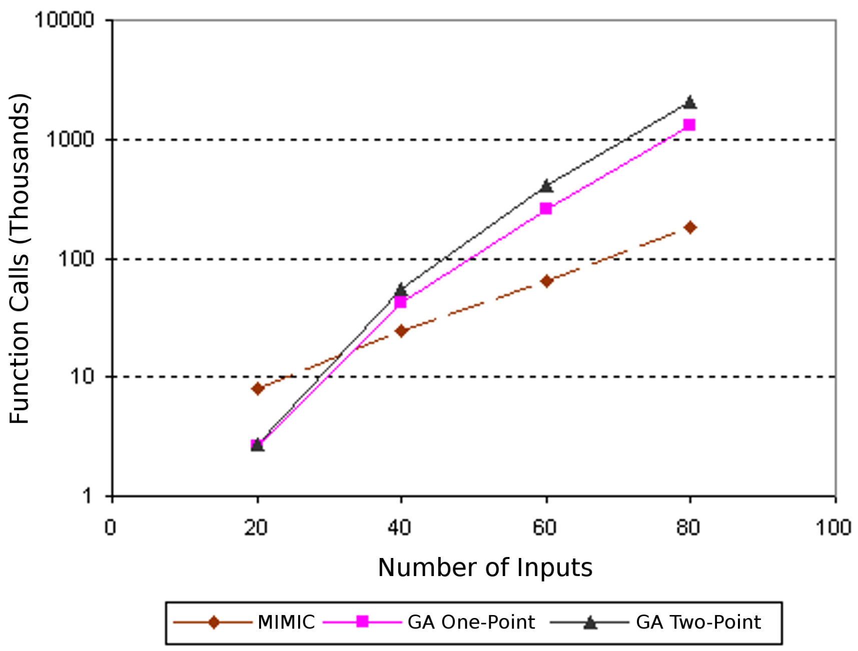


Figure 33: Function Calls Required to Find the Global Optimum.

have on the Four Peaks Problem. For small problem sizes, the GA performed well. However, explicitly modeling the pair-wise inter-dependencies between variables allowed MIMIC to perform more efficiently as the problem size increased. MIMIC obtained an order of magnitude improvement when working with 80 inputs, Figure 34. The mean performance, for each algorithm, matched well with the behavior seen in the work of DeBonet et al.[8]

The points in Figure 33 are bounded averages computed with 95% confidence for all three methods, see Table 2 for computed confidence intervals. For a single run, function calls were tracked until the method reached one of the two global optima.

As the problem size increased, the MIMIC algorithm required a larger sample from the domain to model the pair-wise dependencies. The sample used to build each model was comprised of 300 members for the 20 and 40 input cases. A group of 500 was used for each 60 input case and a group of 1000 was used to model the domain at 80 inputs. The total function call count still strongly favored MIMIC as only an average of 181 iterations where required for 80 inputs as opposed to the 20667

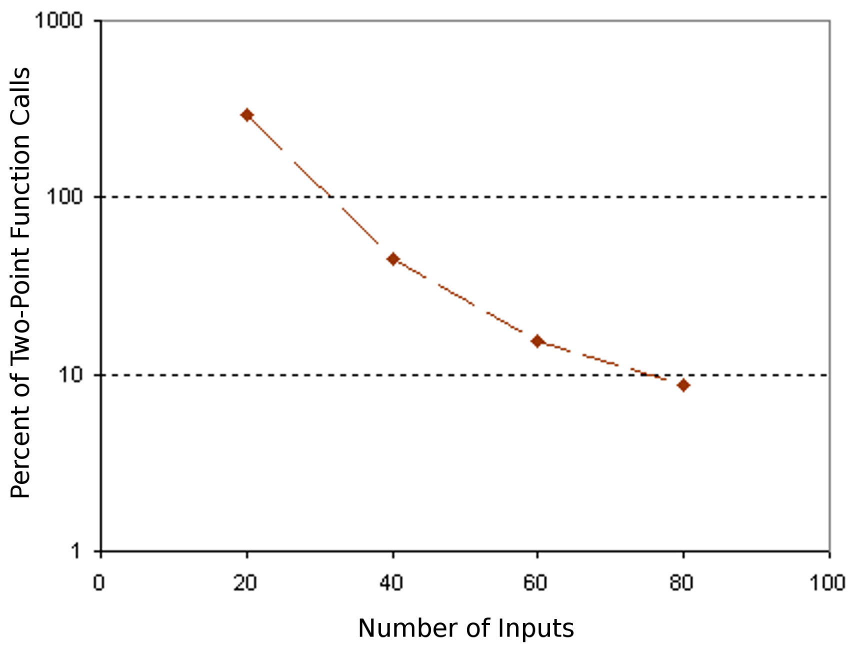


Figure 34: MIMIC Function Calls as a Percentage of Two-Point Function Calls.

iterations required to converge using two-point crossover, Table 2.

Table 2: Bounds for Average (Thousands of Function Calls) Computed with 95% Confidence.

	MIMIC		GA One-Point		GA Two-Point	
Inputs	Avg	+/-	Avg	+/-	Avg	+/-
20	8.04	0.55	2.61	0.15	2.74	0.15
40	24.55	0.70	41.77	1.60	54.72	2.50
60	64.77	1.20	260.33	9.00	415.58	15.00
80	181.56	3.20	1298.00	38.00	2066.73	70.50

The one-point crossover operator had the good fortune of having one ‘cut-point’ correctly placed at the first variable, aiding it to potentially separate the two components of this problem with its the second cut. This is likely the reason for its advantage here over the two-point crossover operation. The number of calls required by MIMIC to create a probabilistic model of the space allows GAs to outperform MIMIC on small versions of this problem. The pair-wise models become more useful, in this problem, as the number of variables increase and allow for solutions at a tenth the cost of the two-point crossover. Since many heuristics use GAs to perform static decomposition, MIMIC could have great impact to the field as a drop in replacement for GAs on large coupled problems.

The four peaks problem described a coupled multimodal domain that allowed for adjustment of problem size to measure the scalability for three methods (MIMIC, GA with One-Point Crossover, and GA with Two-Point Crossover). This problem shows the potential advantages for using a probabilistic model to approximate the distribution of the solution domain, automatically grouping inputs to leverage this information between iterations. In the practice of decomposing a set of analyses, the correct arrangement of analyses using a user specified metric could be found in a similar manner.

One strong source of flexibility for MIMIC is that solely a distribution is passed between iterations. Though the distribution here was initialized as uniform, nothing in the method prevents a user from pre-conditioning the distribution. This quality of MIMIC has application to a wide variety of aerospace problems. Generally speaking, knowledge of the physics of an aerospace problem could be used to develop a reasonable approximate model until the dependencies are better known. A designer could then apply MIMIC to this approximated model. When the method has developed a dependency model for the approximated domain, the evaluation function could be switched to the actual domain. The distribution would then adapt to the true domain, having been assisted by its prior analysis of the approximated problem.

2.3.2 Extension to MIMIC with Continuous PDF Estimation

The MIMIC algorithm requires calculation of mutual information between pairs of variables to approximate the joint distribution between all inputs. The estimated joint distribution provides likely locations for higher performing combinations of inputs. The calculation of mutual information between two variables A and B requires knowledge of three probabilities; P(A), P(B) and P(A,B). MIMIC has been presented as addressing discrete inputs. This allows for these input probabilities to be treated as histograms where entries are counted and normalized by the total number of entries.

The calculation of mutual information in Section 2.1 was shown to be possible with either continuous or discrete random variables. A dependence tree (as in MIMIC) can be used to approximate the full joint distribution between all the input variables once the mutual information is known.[18] An approximation for the continuous probability distributions: P(A), P(B), and P(A,B) is required to apply MIMIC to continuous variables. In this work, a non-parametric method is utilized to estimate continuous probability distributions.

A method is non-parametric if no knowledge of the true distribution is required before estimation. One non-parametric method utilized for an extension to MIMIC is Parzen-window density estimation. Parzen's work [73] allowed for an estimated distribution that would converge toward the true distribution as the number of samples increased. The method has been successfully used in pattern recognition[25], image restoration[3] and regression[52].

$$\begin{aligned} P &\approx \frac{k}{N} && (13) \\ P &= \int_{R^d} p(\bar{x})d\bar{x} \approx \underbrace{p(\bar{x}) \int_{R^d} d\bar{x}}_{\text{small region } R^d} = p(\bar{x})V && (14) \\ p(\bar{x}) &\approx k/NV && (15) \end{aligned}$$

In Equations 13 and 14, P is the probability of a point x existing within a region R. Given N samples independently drawn from an unknown distribution $p(x)$, if k samples fall within R an estimate for P would be shown in Equation 13. Assuming that the region R is small enough that $p(x)$ does not change greatly within it, P could also be approximated by $p(x)$ times the volume of the region (Equation 14). These two pieces provide an estimate for the probability density function p(x) in Equation 15.[39]

$$p(\bar{x}) \approx k/NV = \underbrace{\frac{1}{Nh^d}\sum_{i=1}^{N} K\left(\frac{\bar{x}-\bar{x}_i}{h}\right)}_{\text{h is edge for cube of dimension d}} \tag{16}$$

The value k which represents the number of samples appearing within the region R is often calculated by applying a function K at each sample point, Equation 16. This function K is referred to in machine learning as a kernel function. Depending on the kernel selected, each sample point has an opportunity of contributing to the probability estimate within the region. The region R for simplicity is considered as a hypercube with an edge length of h. In Parzen-window density estimation, the parameter h is referred to as the bandwidth or window-width parameter.

$$K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x_i-x-h\mu_x)^2}{h^2\sigma_x^2} + \frac{(y_i-y-h\mu_y)^2}{h^2\sigma_y^2} - \frac{2\rho(x_i-x-h\mu_x)(y_i-y-h\mu_y)}{h^2\sigma_x\sigma_y}\right]\right) \quad \text{where } \bar{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{17}$$

Many kernel functions are available but the one selected for this work is the Gaussian kernel. The selection of this kernel makes a weak assumption that the true distribution is smooth. Data distributions meeting this assumption are more easily modeled by the Gaussian kernel. Uniform and other non-smooth distributions will still be modeled by the kernel given a sufficient, potentially large number of data points. Convergence to any arbitrary pdf function is possible.[6] The assumption is made in this work that the pdf function for continuous data passed between analyses can be well approximated by a smooth function.

$$K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) = \underbrace{\frac{1}{2\pi}\exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2h^2}\right)}_{\text{where } \mu_x=\mu_y=0,\ \sigma_x=\sigma_y=1,\ \rho=0} \tag{18}$$

The full kernel for a 2-D Gaussian is presented in Equation 18. The model is simplified by using a standard Gaussian with $\mu = 0$ and a $\sigma = 1$ for each point, as in Equation 18. Each Gaussian is centered on its sample point and the window-width h appears in the final equation such that modifications to it behave as if changes were being made to σ, see Equation 19.

The selection of the kernel is not as important as the selection of the window-width parameter h. Setting it too small will overfit the data, while making it too large will underfit the data. A method from the work of Botev [9] is utilized to dynamically select for the parameter h.

$$\begin{aligned} p(\bar{x}) = p\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = p(x,y) &\approx \frac{1}{Nh^2}\sum_{i=1}^{N} K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) \\ p(x,y) &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\pi\, h^2}\exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2h^2}\right) \end{aligned} \quad (19)$$

Parzen-window estimation for $p(x,y)$, using a Gaussian kernel, requires the use of Equation 19 over all available samples. The required estimation of p(x) and p(y) needed to calculate the mutual information between continuous variables can be calculated by using a 1-D Gaussian kernel and results in Equation 20.

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{2\pi}\, h}\exp\left(-\frac{(x-x_i)^2}{2h^2}\right) \quad (20)$$

This can be intuitively considered as a summation of Gaussian distributions, one placed at each sample point, that is then normalized to create a pdf estimate, Figure 35.

By using Parzen-window density estimation, the components needed by MIMIC for continuous structural modeling can be found. This allows MIMIC to be applied to both continuous and discrete variables. The use of kernel density estimation, specifically Parzen Window Estimation, to calculate probability densities allows for

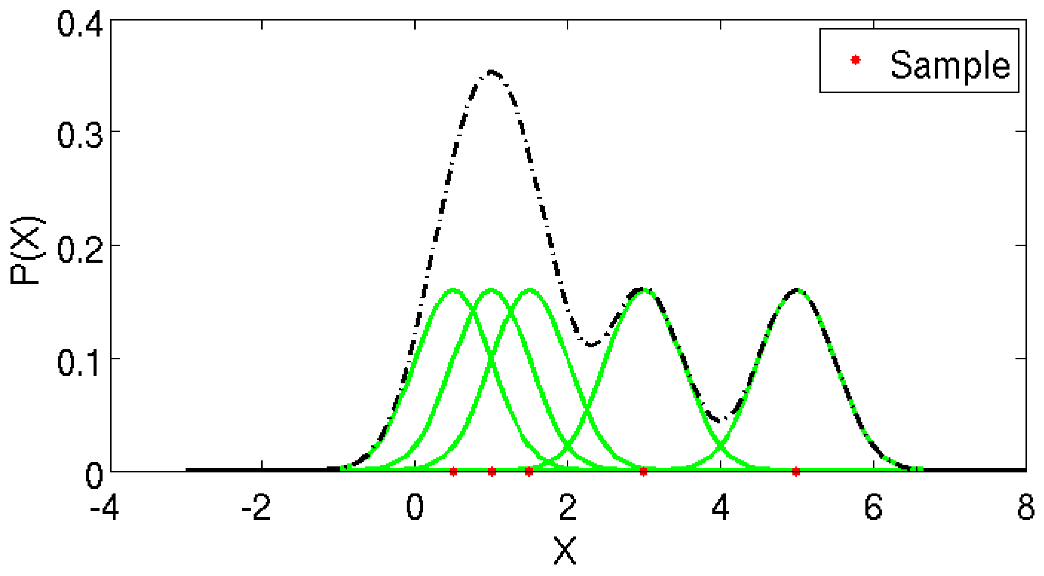


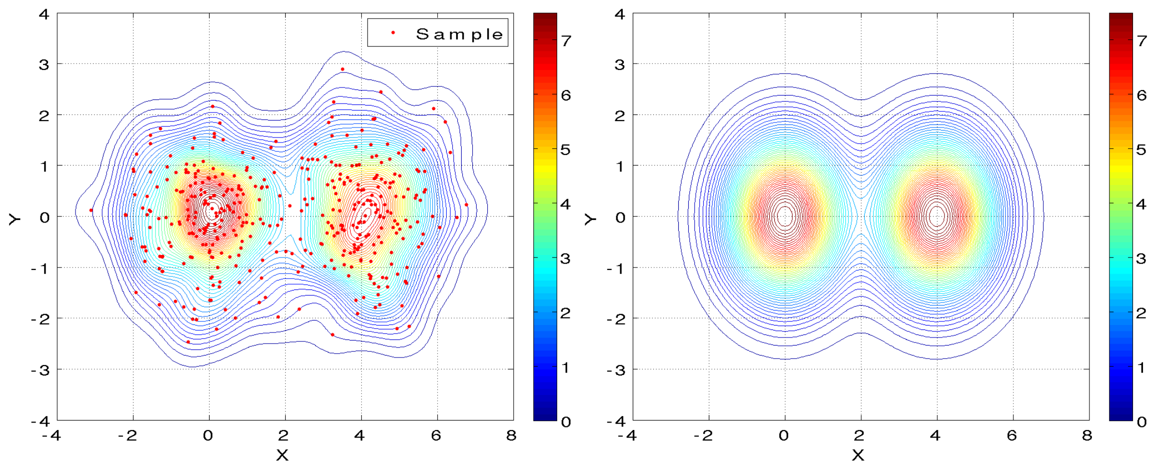
Figure 35: Approximation from Five Samples using Gaussian Kernel. (h=0.5)

the estimation of mutual information between continuous variables.[52] A demonstration that calculates the mutual information between two continuous random variables using kernel estimated probability functions is shown in Section 2.3.2.1.

2.3.2.1 Example of Kernel Method Continuous PDF Estimation

The continuous marginal and pair-wise joint distributions of input variables must be modeled in order to use MIMIC with continuous variables. The modeling method described here is used to approximate continuous distributions using a finite set of sample data. The mutual information between continuous and/or discrete variables can then be solved numerically and utilized by the MIMIC algorithm.

This is a demonstration that calculates the mutual information between two continuous random variables that were estimated first using kernel probability functions. This is the crucial step required for MIMIC to operate on continuous variables. Continuous random variables need to have their marginal and pair-wise probability functions estimated with the information then used to calculate their mutual information. The rest of the MIMIC algorithm would treat both discrete and continuous variables equivalently.



(a) Kernel Approximation (b) True Distribution

Figure 36: Kernel Probability Estimation of P(X,Y) using 400 random samples.

Four hundred samples from the distribution in Figure 36(b) were taken and Parzen-window density estimation was used to generate the approximate multimodal density function shown in Figure 36(a).

Table 3: Information Theory Attributes for Estimated Variables X and Y

	Entropy (Est)	**Cross Entropy with Y (Est)**	**Mutual Info with Y (Est)**	**Mutual Info Actual**	**Diff**
X	13.251	14.324	11.721	12.180	-3.77 %
Y	12.793				

These approximations were taken to estimate the mutual information between the two variables, shown in Table 3. This was compared against the actual computed mutual information between the two variables. The proximity of the two results indicate that Parzen-window density estimation shows potential for modeling continuous variables for MIMIC. This will allow the use of MIMIC on the many continuous and mixed multimodal problems examined in engineering.

2.3.3 Extension of Results through Decision Tree Learning

2.3.3.1 Decision Tree Learning

There are several methods available in Machine Learning to automatically generate decision trees based on a set of training data. The method that was used in this thesis is called J48 and is an extension on the older C4.5 classification algorithm.[75] The variable that provides the greatest normalized information (in the information theory sense) towards determining a classification is placed highest in the tree. The training set is separated into instances on either side of the tree branch and the process is repeated to select the variable that will separate the remaining data sets for the greatest gain in information. Seeking the greatest gain in information at each step is equivalent to taking choices in the decision tree to lower the remaining uncertainty in the answer by the greatest amount at each step. Choices that greatly reduce the uncertainty regarding the final classification will appear earlier in the tree, reducing the averaged time for the tree to return a classification. Here the variable with the greatest mutual information to the output variable would reduce the greatest amount of uncertainty in the output variable when it's value is set.

Ten-fold cross-validation was used to validate the discovered decision tree. The data set was randomly separated into ten segments, allowing nine for use as a training set while one was retained for the validation set to the model. Each possible combination of the 10 sets into nine training sets and one validation set was then examined to provide a better averaged estimate for the performance of the decision tree to unseen data.[63]

To illustrate the use of decision tree learning, a planetary entry example is utilized.

2.3.3.2 Introduction to Decision Tree Example Dataset

The PESST framework described in Chapter 5 was used to simulate an entry body at Venus over several ballistic coefficients (BC), entry flight path angles (FPA) and entry velocities. The output of concern was whether the vehicle would skip out of the atmosphere and how the variables would rank as drivers for this process.

The Pioneer-Venus probes were sent in 1978 to gain information regarding the Venus entry environment.[85] The entry domain used for this study includes the entry values for the shallower two Pioneer-Venus probes. The cases for Venus entry covered the entry conditions utilized by the Pioneer-Venus Sounder and Day probe; both had an entry velocity of approximately 11.54 km/s and an entry flight path angle of -32.37°(Sounder) and -25.44°(Day).[86] The cases examined entries with ballistic coefficients at 150, 200 and 250 kg/m^2 both probes had a ballistic coefficient of approximately 200 kg/m^2.

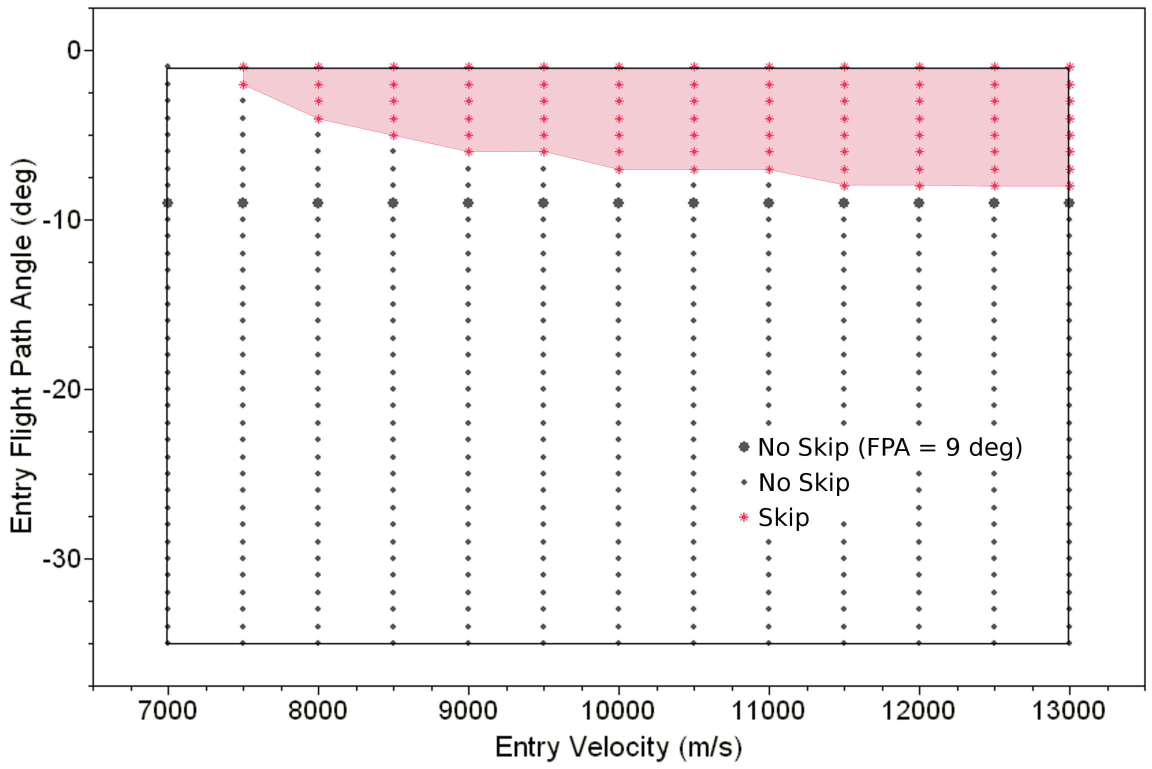


Figure 37: Design Space for a Venus Skip (BC = 200 kg/m^2).

When a variable has a uniform distribution, the entropy associated with the variable is a function of the number of potential values it can take. Ballistic coefficient, entry velocity and entry FPA were all equally distributed in a full factorial experiment over the domain. The output boolean variable for skipping has a lower entropy than a 50/50 choice as the domain had a higher percentage of non-skipping cases. The important measurement here is the amount of shared entropy between the input and output variable, shown by the calculated mutual information. The mutual information is highest for those variables that provide the greatest amount of information regarding the value of the output; entry FPA is the most important variable regarding skip out, then entry velocity and finally ballistic coefficient. The results returned did not show any difference in skip out status due to the ballistic coefficient making these results independent over the Venus domain specified. This result says nothing regarding heating, lowest altitude attained, or time required to exit; only that the final fact of the vehicle leaving the atmosphere was not effected by ballistic coefficient over the domain specified.

2.3.3.3 Demonstration of Decision Tree Generation

This work develops descriptive metrics to predict performance over a region of the design space instead of only showing static decomposition methods applied to specific problems. After the generation of data sets covering the comparison between MIMIC and GA on a number of different DSM problems, machine learning methods are utilized to generate decision trees. These trees provide guidance to a user regarding which method should be used on similar problems.

The two strongest variables affecting skip out for a range of Venus entries was shown to be entry FPA and entry velocity. The relationship of these variables to skip out over the domain is shown graphically in Figure 37. No cases were shown to skip out for the domain examined below negative nine degrees. Numerically Table 4 shows that entry FPA shared the greatest amount of information with the output variable for skip out.

Table 4: Information Metric Applied to Venus Entry Skipping

	Entropy	**Cross Entropy with Skip**	**Mutual Information with Skip**
Ballistic Coefficient	1.585	2.236	0.000
Entry Velocity	3.700	4.317	0.035
Entry Flight Path Angle	5.129	5.289	0.492
Skip	0.651		

Provided with these inputs and asked to create a decision tree, the input with that provides the greatest amount of information would be selected as the root node for the tree. In this case, the entry flight path angle provides the most information regarding whether a Venus entry will skip over the domain of interest. After assigning a range of values to the flight path angle, the variables are reexamined to discover which would most greatly reduce the uncertainty towards knowing if the entry skipped.

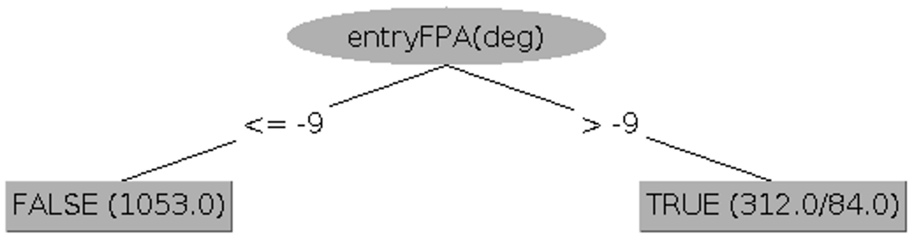


Figure 38: Automatically Generated Decision Stump for Venus Skipping.

A decision stump for the data set is shown in Figure 38. A decision stump is composed of one choice that separates a group of cases into two sub-groups. The variable selected for the choice node is selected automatically from the input variables. The variable with the greatest ability to separate the cases into classes is selected as the node that provides the largest reduction in uncertainty associated with the answer. Referring back to the earlier discussion on information theory, this variable provides the greatest information gain when its value is known. The first variable selected has the greatest mutual information with the output classification.

The automatically created stump selected entry flight path angle first as it has

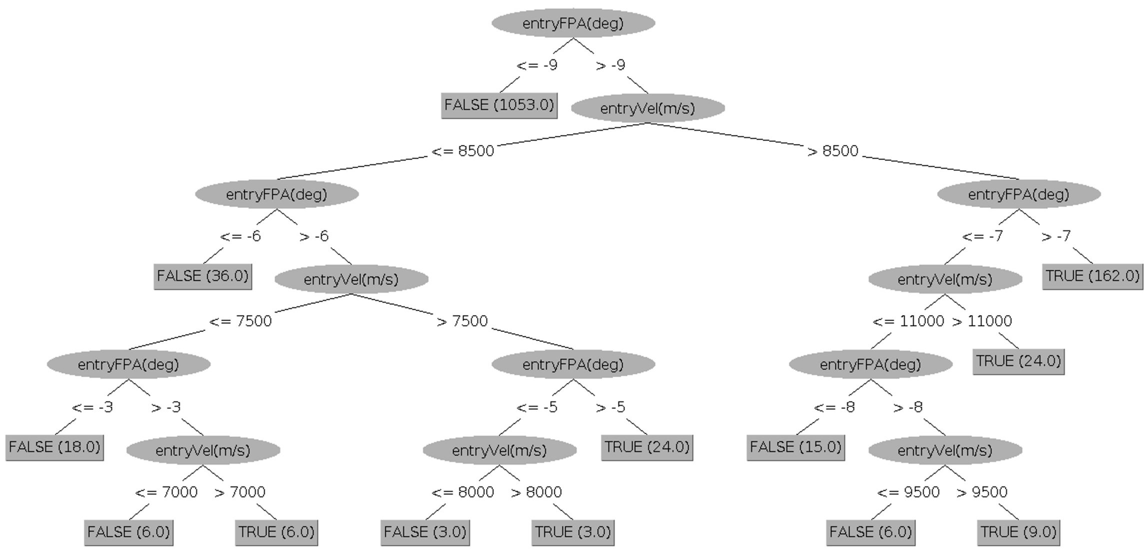
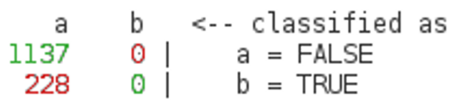


Figure 39: Automatically Generated Decision Tree for Venus Skipping.

the greatest amount of mutual information with whether the vehicle will skip out of the atmosphere, see Table 4. The next branch for the tree would be selected with the value of the next most important input variable; specified by the one with the next highest amount of mutual information with the output.

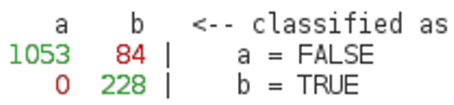
One last important remark on the decision stump in Figure 38 is that the value used to separate the cases into two branches was selected by the algorithm as 9 degrees. The chart shown in Figure 37 demonstrates that this is an excellent value to separate two segments of the entry design space. The value separates over 80% of the design space that did not skip from the other segment that has a probability of skipping. This shows that the automated process is able to develop reasonable values for continuous input variables during the creation of a decision tree. This decision stump was able to correctly classify 93.8% of the cases during 10-fold cross-validation.

Allowing the tree to generate sub-branch structure allows for the method to seek a more complicated representation to better classify the cases presented as training data. The larger tree in Figure 39 is able to correctly classify 99.9% of the cases as measured by 10-fold cross-validation. This is compared against the accuracy of a



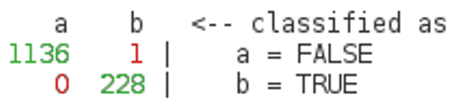
```
     a     b   <-- classified as
  1137     0 |     a = FALSE
   228     0 |     b = TRUE
```

(a) Single Rule (83.3%)



```
     a     b   <-- classified as
  1053    84 |     a = FALSE
     0   228 |     b = TRUE
```

(b) Decision Stump (93.8%)



```
     a     b   <-- classified as
  1136     1 |     a = FALSE
     0   228 |     b = TRUE
```

(c) Decision Tree (99.9%)

Figure 40: Accuracies for Three Classification Methods

simple rule to always guess the most likely classification 'Does not Skip' in Table 40.

The comparison between a decision stump and a larger decision tree demonstrates how additional accuracy can be obtained at the cost of increasing model complexity. An appropriate balance between model complexity and predicted performance can be balanced by modifying the amount of branch pruning allowed by the method. The amount of branch pruning applied to decision tree generation effects the complexity of the resulting decision tree by setting the minimum number of cases at a leaf node required before generating a new branch.

The matrices in Table 40 describe the number of cases that are correctly classified along the horizontal, the number of category A that are correctly classified as category A. The two off diagonal locations in red show the number of incorrectly classified cases. Category A is used for 'Does not Skip' while B is used for 'Does Skip'. A single rule to always pick the most likely option in the domain obtains an accuracy of 83.3% in Figure 40(a) while a decision stump is able to increase this accuracy to 93.8%. The stump is able to provide a good deal of guidance while not being a complicated structure. This serves as a proof of concept that a decision tree can be created with an input data set, its accuracy and complexity balanced to serve as a guide to users. The method of generating decision trees based on generated data sets is flexible and could be used to provide guidance in generalizing DSM comparison results to a larger portion of the design space.

CHAPTER III

IMPORTANCE HEURISTIC: EVALUATION FOR ENGINEERING

The value of mutual information as an importance heuristic for engineering design is examined first on a realistic first order design tool developed as part of this work. The design tool will allow the examination of the heuristic on a trade study for a planetary entry problem. Mutual information is compared with correlation through their use in understanding the trade study.

3.1 Validation of Link Rank Heuristic

The link importance metric proposed by this work, mutual information will be examined in this section and validated on a planetary entry problem. The planetary entry problem was examined through the creation of the PESST framework that was created as part of this work (described in Chapter 5).

3.1.1 The PESST Framework

The Planetary Entry Systems Synthesis Tool (PESST) was developed to be a usable conceptual design tool for spacecraft entry studies of Earth, Mars and Venus.[71] The tool incorporates discipline models for geometry, hyper-sonic aerodynamics, guidance algorithms, trajectory simulation, thermal environment and sizing to converge conceptual entry vehicles. The majority of PESST was written by the author with excellent analysis contributions in guidance from Brad Steinfeldt and Michael Grant; Patrick Smith later contributed parametric analysis functionality to automatically run PESST through a user specified design space. A detailed discussion of the models utilized for PESST are included as part of this thesis work in Chapter 5.

For the demonstration experiment, the PESST tool is used to analyze a Martian entry design space. The ranks for several variables over the design space will be computed based on the mutual information that the variable provides towards objective. The entries are classified by whether they were able to successfully reduce their speed to below 1 m/s. Each variable over the design space was examined as to how well a knowledge of that variable, at an altitude, would have increased the knowledge of whether the vehicle would be able to meet its velocity target. Variables that have only one value per trajectory (e.g. Total Propellant Mass) would have constant mutual information values over the trajectory. Variables that change during the trajectory (e.g. Velocity) could see their mutual information with the targeted objective change.

In the case where several variables are ranked for a single link, the highest ranked variable passed in the link will be used to estimate the importance of the link as a worst case estimate. Variables that have a vector of values for each trajectory will have the maximum absolute value used to represent a worst case.

3.1.2 Validation on Planetary Entry Problem

The PESST framework was used to validate using mutual information as a method of ranking DSM links for a conceptual design study. This is not an analysis of what variables are important to all EDL missions. It focuses on one design study explained below and ranks the DSM links based on the returned data. The same technique could be used over other design spaces to rank link importance.

A list of input variables and inter-disciplinary variables were formed in order to have coverage over most of the links of the PESST DSM for the study. A Mars entry domain was specified that spanned different entry conditions and sphere-cone geometries, Figure 5. The evaluation of the mutual information metric will then be compared against the use of correlation to rank the same variables. The advantages and disadvantages to the method are discussed after displaying study results.

Table 5: Bounds on Entry Design Space

Input Variable	**Lower Value**	**Middle Value**	**Upper Value**
Atmosphere (Temp, Density, Pressure)[1]	Low	Aver	High
Nose Radius (m)	0.5	0.65	0.8
Cone Half Angle (°)	50	65	80
Entry Velocity (m/s)	7500	8500	9500
Entry Flight Path Angle (°)	-13	-15.5	-18
Maximum Thrust of Engines (N)	5000	10000	15000
Isp of Engines (s)	300	310	320
Initial Temp of TPS (°C)	-85	-75	-65
Number of Cases Examined	$3^8 = 6561$		

The 6561 cases from the design domain in Figure 5 are all examined to determine if they were able to reduce their velocity to less than 1m/s at an altitude of -2500m. The details for the entry event time-line and triggers follow in the next section.

3.1.2.1 Planetary Entry Problem Events and Tracked Variables

The domain limits in Figure 5 were selected to generously bound the entry conditions and geometry selections for a general direct entry Mars mission with a final gravity turn performed by liquid-based propulsion. The event framework in PESST works by the specification of a start and completion trigger for events that span a period of time. Events, as a heatshield drop, require only a single trigger. Many options are available for the triggering variable but only one variable value can be used for triggering an event. Parachutes, for instance, could be triggered by Mach number or dynamic pressure but not by a combination of both variables.

The events specified over the domain space and the event triggers that have been selected are shown in Figure 6. The parachute uses a Mach trigger as the value shown

[1]Low, Average and High atmospheric profiles were produced through Mars GRAM for an entry at latitude 22.63°and longitude 338°(Pathfinder entry location) during October 5, 2011.

is nearing the limits of current parachute technology. Relative triggers are available so the heatshield can be dropped a number of seconds after parachute deployment. The gravity turn seeks to target a final velocity of less than 1m/s at −2500m. It specifically targets a velocity of 0m/s but any velocity under 1m/s was accepted as near enough.

Table 6: Events Programmed to Occur During Entry

Event	**Start Trigger**	**End Trigger**
Parachute Deployment	Mach = 2.5	Altitude = −1500m
Heatshield Drop	Drop 5 seconds after parachute deploys	N/A
Gravity Turn	Start 1 second after parachute event ends	Altitude = −2500m targets velocity = 0m/s

The events for this entry are closely interrelated which is realistic. If one event fails to trigger other closely connected events will likely also fail to trigger. For the events in Figure 6, if the trajectory never falls below Mach 2.5, then the parachutes will never open. This would cause the heatshield to fail to drop as the event is triggered 5 seconds after parachute deployment. Without the parachute event, the gravity turn event would fail to start due to the fact that the parachute event never ended.

A number of variables were tracked during the analysis over the design space, Figure 7. These variables included design study variables, converged subsystem masses, variables that change over the course of the trajectory and variables dealing with the thermal protection system. The selection of variables spanned most of the links in the PESST DSM and were recorded 0.1 second intervals along the trajectory.

The variables from Figure 7 were recorded at specified altitude marks to provide a stable frame of reference from which to judge variable values over the design space. Converged total system masses do not change over the course of a trajectory. The actual mass present at the vehicle will change with time but the converged total parachute system mass at entry will not change given a trajectory.

Table 7: Variables Tracked in Design Study

Design Variables		
Cone Half Angle	Entry FPA	Entry Velocity
Initial Temp of TPS	Isp of Engines	Max Available Thrust
Nose Radius	Planet Atmosphere	
Mass Breakdown		
Backshell Mass	Entry Mass	Heatshield Mass
Landed Mass	Parachute Mass	Propellant Mass
Propulsion Sys Mass		
Thermal Protection System		
Recession Thickness	TPS Thickness	
Trajectory Variables Sampled with Respect to Time		
Altitude	Angle of Attack	Azimuth
Ballistic Coefficient	Bank Angle	CD
CdA	Convective Heatload	Convective Heatrate
Crossrange	Density	Downrange
Drag	Dynamic Pressure	Flight Path Angle
Latitude	Lift/Drag	Longitude
Mach	Radiative Heatload	Radiative Heatrate
Sensed Deceleration	Thrust	Time
Total Heatload	Total Heatrate	Vehicle Mass
Velocity		

The variables that only have one value per trajectory will have constant mutual information. In this work, altitude was used to decide the point of comparison over the design space. Mach number or time could have been selected as well. To compare consistently over all points in the design space, the user should select a variable to use as a reference that appears in all domain cases. If half the points in the design space never reach Mach 2 then only the mutual information of the remaining values can be computed at Mach 2.

3.1.2.2 Evaluation of Proposed and Standard Metric

A comparison was performed between correlation and mutual information on the same trajectory data. A very common from of correlation used in engineering, Pearson's correlation, measures the linear dependence between two variables. The tendency of two variables to vary their values in either the same direction (positive number) or opposing direction (negative number). There is directional information that is provided by the sign of the number provided. Mutual information provides a measure of the non-linear dependence of two random variables but there is no concept of slope or direction to the number provided. The mutual information will always be zero or higher, reflecting the number of bits that could be used to describe the information shared between the two variables.

Table 8 was formed by calculating the mutual information and correlation between each variable and the system objective along the trajectory. The largest dependence recorded was used as the variables entry in the table.

The values for all trajectory variables were taken at 15 set points along the trajectory. Altitude was used to set these points at 110km, 90km, 70km, 50km, 40km, 30km, 20km, 10km, 5km, 3km, 1km, 0km, −1km, −2km and −2.5km. The drawback to selecting a reference variable in this way is that changes in the variable can't be measured to determine either its correlation or mutual information to the objective.

Beyond numerical noise, all the values being measured for altitude in this case are equal, during one batch of trajectory sampling, regardless of the final ability to meet the velocity objective. For mutual information, altitude would therefore provide no knowledge to the trajectories ability to meet the final goal making the mutual information equal to zero. The value should be zero or, with numerical error, close to zero for correlation. This is seen to be so for both cases in Table 8.

The maximum value obtained by the variables during their trajectory was used for Table 8. The top six variables were plotted as a function of altitude to show their behavior over the trajectory. Variables that only have one value for each trajectory (e.g. total propulsion mass) have a constant value on the graph. Variables that contain a vector of values for every trajectory are plotted as a function of altitude to show how the mutual information calculated between the variable and the system target change over the coarse of the trajectory.

Table 8: Ranked Variables for a <1m/s Target at -2.5km

Max MI	Label	Max Corr	Label
0.8614	Velocity	0.5758	Total Propellant Mass
0.8195	Drag	-0.51	Azimuth
0.5595	Dynamic Pressure	-0.4498	Max Available Thrust
0.5551	Thrust	0.3751	Thrust
0.5462	Total Propellant Mass	-0.3471	Velocity
0.5260	Total Propulsion System Mass	-0.3471	Mach
0.3962	Azimuth	-0.3016	Vehicle Mass
0.3182	Flight Path Angle	-0.281	CD
0.2570	Time	0.2803	CdA
0.2004	BC	-0.2802	BC
0.1742	Landed Mass	-0.2793	Dynamic Pressure
0.1623	Max Available Thrust	-0.2791	Sensed Deceleration
0.1598	Vehicle Mass	-0.279	Drag
0.1489	Total Heatload	-0.2742	Convective Heatrate
0.1340	Convective Heatload	-0.2742	Total Heatrate
0.1302	Convective Heatrate	-0.2719	Flight Path Angle
0.1300	Radiative Heatload	0.2679	Entry Mass

Table 8: Ranked Variables for a <1m/s Target at -2.5km

Max MI	Label	Max Corr	Label
0.1260	Downrange	-0.1757	Parachute Mass
0.1198	Total Heatrate	-0.1643	Radiative Heatrate
0.1181	Crossrange	0.162	Time
0.0970	Parachute Mass	0.1442	Cone Half Angle
0.0961	Entry Mass	-0.1394	Total Propulsion System Mass
0.0889	Mach	-0.0776	Radiative Heatload
0.0874	CdA	0.0742	Altitude
0.0853	Sensed Deceleration	0.0694	Entry FPA
0.0727	Radiative Heatrate	-0.0694	Longitude
0.0314	CD	-0.0665	Heatshield Mass
0.0251	Heatshield Mass	0.0654	Landed Mass
0.0218	Cone Half Angle	0.0636	Downrange
0.0092	Longitude	-0.0625	Latitude
0.0058	Entry FPA	-0.0602	Crossrange
0.0058	Latitude	-0.0495	Recession Thickness
0.0021	Planet Atmosphere	-0.0407	Density
0.0013	TPS Thickness	-0.0391	Planet Atmosphere
0.0010	Recession Thickness	0.0354	Convective Heatload
0	Nose Radius	0.0354	Total Heatload
0	Entry Velocity	-0.0149	TPS Thickness
0	Isp of Engines	-0.0107	Nose Radius
0	Initial Temp of TPS	-0.0081	Entry Velocity

Table 8: Ranked Variables for a <1m/s Target at -2.5km

Max MI	Label	Max Corr	Label
0	Backshell Mass	-0.0069	Initial Temp of TPS
0	Altitude	-0.0012	Isp of Engines
0	Lift/Drag	0	Backshell Mass
0	Density	0	Lift/Drag
0	Angle of Attack	0	Angle of Attack
0	Bank Angle	0	Bank Angle

No information is gained by the value of thrust during the early segment of the trajectory that would help determine if the system reaches its final velocity target. That is why the mutual information between the thrust variable and the velocity target is zero during the early segment of the trajectory. Knowing the value of thrust at 50km tells one nothing about whether the final velocity target will be reached, see Figure 41.

The total propellant mass and total propulsion system mass are design drivers due to the importance of the gravity turn event. The gravity turn event was triggered to start one second after the parachute event ended. The parachute is told to trigger at Mach 2.5 and end at an altitude of -1.5km. If the vehicle fails to meet Mach 2.5 before -1.5km then the parachute will open but never hit its release trigger. The parachute will be active for the rest of the trajectory and the guidance event will fail to ever become active.

The parachute changes the aerodynamics of the vehicle strongly, greatly increasing

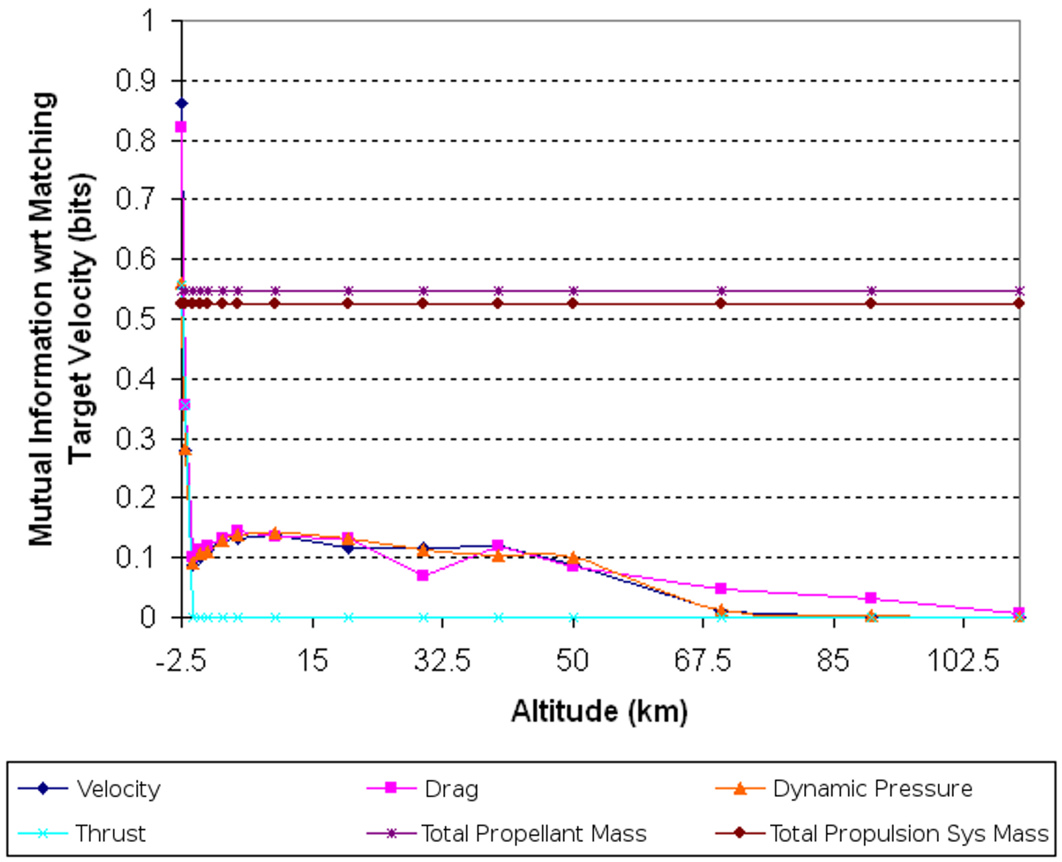


Figure 41: Mutual Information of Variables Sampled by Altitude

the vehicle drag. Cases in the design space exist where the triggers used force the parachute to remain active below its release altitude of −1.5km. The failure of the guidance event to fire will mean a total propellant mass of zero and a much lower total propulsion system mass as the system is told to size tanks for zero propellant.

The value of the sized propellant and propulsion system then dictates whether the system reaches its final velocity target. The importance of drag is indicated by the parachute event straying into altitudes lower than 1.5km. If the value of the drag variable is large at these low altitudes the system is almost certain to not hit its velocity target as the guidance event will not fire.

The system target of whether the vehicle could reduce its velocity to under 1m/s

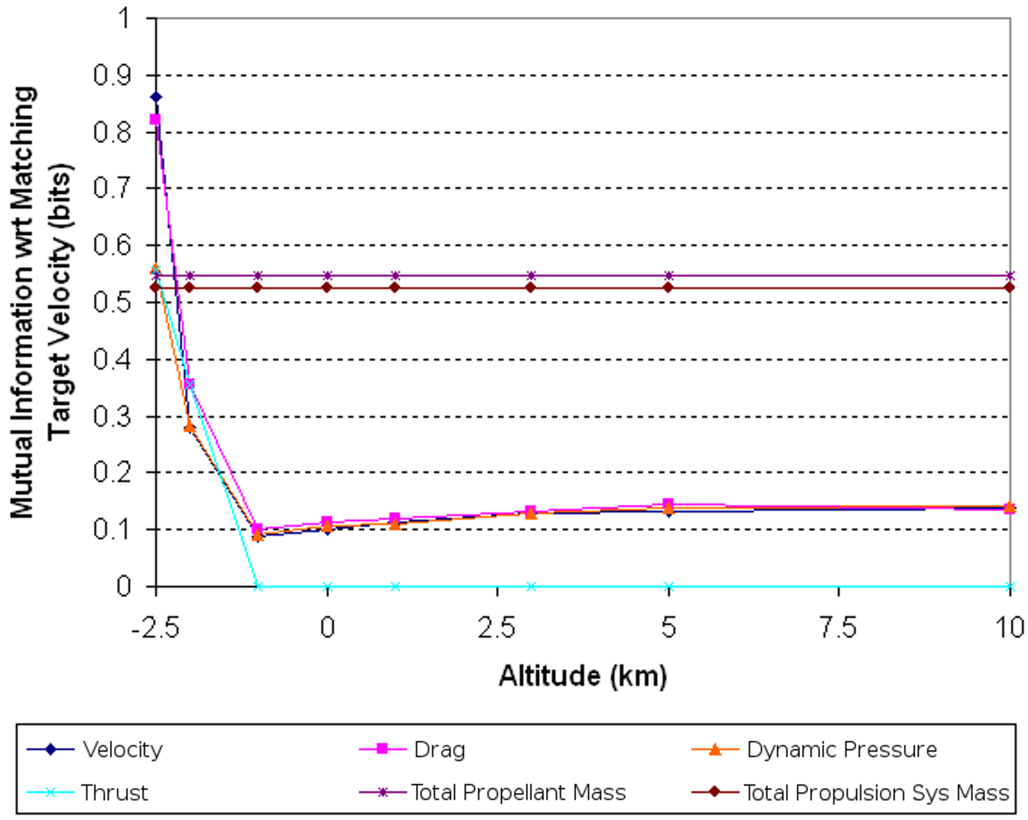


Figure 42: Detail for Mutual Information of Variables Sampled by Altitude

is a binary choice. If the design space had an equal number of successes to failures on this system metric (i.e. 50% yes and 50% no) than this variable would require at least one full computer bit to describe each result in the design space. This is a measurement of the random variable's entropy which is described in the discussion of mutual information (i.e. shared entropy) in Section 2.1. This point is important because it means that the worst case entropy for a binary variable is 1 bit. This means that the largest shared entropy that can be had with a binary variable is equal to 1 bit. Given this, velocity and drag in Figure 42 are both able to almost completely remove any uncertainty regarding the value of system target metric. The system metric is not split in a 50/50 manner over the design space so the target entropy is instead just above 0.9.

These results highlight the importance of mutual information in a realistic conceptual design study. Mutual information provided a measurement of how important each variable was by measuring how much uncertainty the knowledge of each variable would reduce in the target. Examining the graph quickly showed that something important, to our target metric, was happening with the guidance system. Tracing the 'why' behind the importance of drag would reveal the parachute event as being active below its release altitude. This demonstrates its use as a tool for better understanding the results from design studies, allowing an additional use as a debugging tool. Velocity is calculated as the most important variable which makes a measure of intuitive sense as the system target metric is based off of the final velocity.

This is now contrasted with design knowledge obtained by using linear correlation to examine the same design space and points. Correlation provides a direction to the linear relationship that is missing from the mutual information metric but does not perform as well at revealing the relationships and dependencies discovered though mutual information.

In Figure 43, Mach decreases as the chance of obtaining the target velocity increases. The linear slope follows a reasonable direction but it is ranked by mutual information a being the 23rd most important variable tracked for the trajectory with respect to being able to determine if the velocity target is met. There is no question that Mach is decreasing for all trajectories traveling through the atmosphere but this information does not include how important the direction of the relationship is to meeting the system target. The mutual information measure lacks this directional information is it measures both linear and non-linear dependence. Mutual information though measures the amount of the system target uncertainty explained by knowing the value of the other variable.

As the total propellant mass in Figure 43 increases, obtaining the target velocity

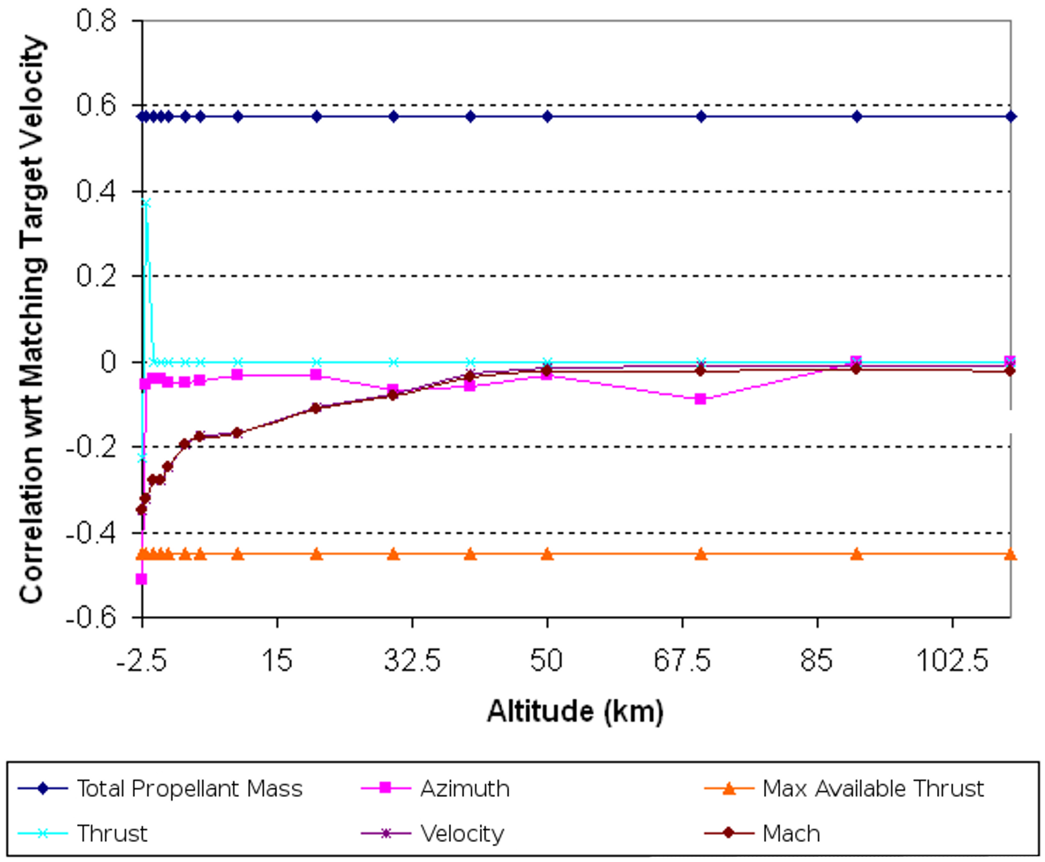


Figure 43: Correlation of Variables Sampled by Altitude

becomes more likely. The listing of the total propulsion mass agrees with the calculation of mutual information but the total propulsion system mass is 22nd in the correlation list even though it is directly affected by a lack of sized propulsion mass. This is another example that shows the difference between correlation and mutual information. Correlation is valuable for determining the direction of relationships but mutual information should be used when one wants to measure how dependent the value of one random variable is on another. The linear dependence of the total propulsion system mass might be low but it is directly dependent on the total propellant mass sized for the system. Mutual information was able to recognize this dependence while linear correlation was not.

Events in the simulation have the opportunity to start and end at discrete intervals (e.g. every 0.2 seconds). The guidance event is set in the study to wait until the maximum available thrust is required to meet the targeted velocity. Given the discretization of the start/end time a lower maximum available thrust would make the constant uncertainties in firing times lead to smaller uncertainties in the final targeted velocity. A 15kN engine firing for an extra 0.2 seconds leads to a larger change in velocity than the same extra time spent with a 5kN engine. Though the direction of the relationship makes sense, the strength of the linear relationship does not translate to a high mutual information ranking. The maximum available thrust appears as 12th on the list of important variables with respect to the obtaining the targeted velocity.

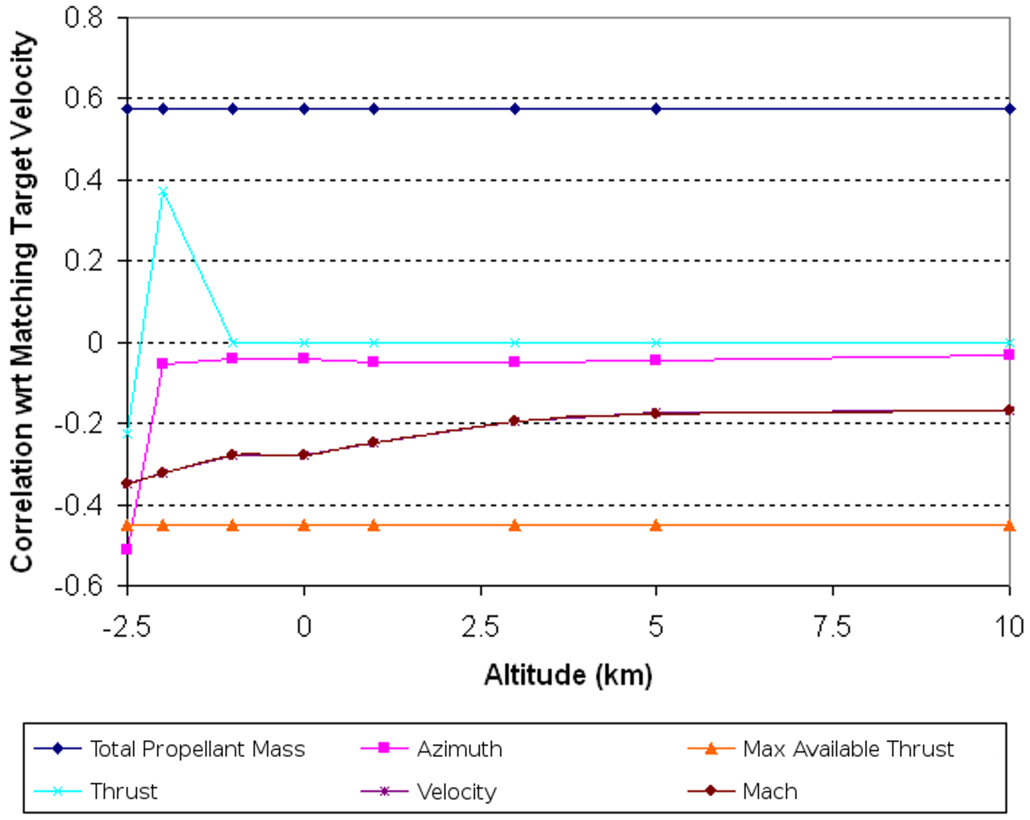


Figure 44: Detail for Correlation of Variables Sampled by Altitude

The importance computed by mutual information corresponds with the correlation strength calculated for thrust. The presence of increasing thrust is directly proportional to obtaining the targeted velocity but then the direction of the relationship changes at the end of the trajectory, Figure 44. This change may be due to the linearization of the relationship or to the thrust value serving as a stand in for the value of the maximum allowable thrust. One group of cases with failed guidance firings would have a thrust value of zero while successfully begun guidance firings would be subjected to time discretization errors that provide finer velocity targeting to vehicles using a lower maximum available thrust. This is a complex relationship that correlation calculates as falling linearly towards a negative correlation. Mutual information does not have difficulty determining that, regardless of directionality, the value of thrust removes a sizable degree of uncertainty regarding whether the system targeted velocity was reached.

3.1.3 Advantages/Disadvantages of Proposed Metric

The computational cost for calculating link importance requires the estimation of a marginal probability distribution and a joint probability distribution. These are formed by tracking and storing values as they are generated by the DSM. As long as the time and/or number of required analyses calls is significantly larger than those required to estimate the probability distributions this is a workable method for determining dependence. For the PESST framework example, a single case takes 15-30 seconds to run meaning several hours for the full domain of data points. The evaluation of the mutual information is on the order of seconds making estimated intermediate rankings workable for this conceptual design tool after a sufficient number of full cases.

This heuristic has a great deal of flexibility toward working with different types of data and can be computed on-the-fly or post-processed after a design of experiments.

An engineer may wish to use this metric as another way to understand the dependence existing between variables in a study, as an alternative to correlation.

Correlation provides valuable information regarding the direction of a linear relationship that is not available through mutual information. Mutual information only focuses on the amount of uncertainty in one random variable explained by knowledge of another variable's value. Something could have a strong correlation without being a major driver that strongly determines a target's value. There is though a strong value to having calculated linear relationships that can be used for estimating the rank of response strength.

3.2 Forced-Based Clustering with Importance Heuristic

Dynamic decomposition involves the discovery and utilization of problem structure information obtained while a problem is solved. While static decomposition uses heuristics to predict future discipline behavior, dynamic decomposition seeks to leverage run-time data to separate a problem into sub-problems. This increases the run-time cost of the decomposition with the benefit of accessing the actual problem's behavior. Heuristics developed for static decomposition (e.g. minimizing feedbacks) may not be the most applicable to the problem being solved. By recording and leveraging problem behavior, problem performance can suggest a tailored decomposition to use.

This work demonstrates a method that can be used with future work on dynamic decomposition and demonstrates the method on a low-thrust trajectory problem. This problem will be decomposed to discover sub-problem structure using force-based clustering with mutual information. As the analyses are connected in a linear fashion, with no feedback loops, common heuristics from static decomposition would find the problem challenging. For example, the problem already has a minimum number of feedback loops and the distance of links from the diagonal of the DSM is already minimized. The only way to select a decomposition is to either blindly split the problem into equal pieces or to rank the importance of the individual links. The clustering here is enabled by assigning each information link with an importance value calculated from the current run-time behavior. Mutual information is proposed, as part of this thesis work, to serve as the appropriate heuristic and force-based clustering is used to determine where the split should occur.

3.2.1 Low-Thrust Problem Domain

The 3rd Global Trajectory Optimization Competition (GTOC) solution presented by the Georgia Tech team was used as an example for trajectory decomposition. The decomposition method is not designed to determine which combination of asteroid targets should be visited by a vehicle. The method presented is geared to discover how a given trajectory should be separated into sub-problems in order to reduce the negative impacts of decomposition (e.g. solving sub-problems that do not help to solve the original problem).

Selection of the solution determined by the team allows for the knowledge that a solution for the problem exists. No information was known a priori on where the trajectory would be best cut to minimize computational time. The trajectory, Figure 45, involves a rendezvous at three asteroids or at least 60 days. Two flyby passes of Earth occur before a final rendezvous at Earth. The same ballistic guess was used for solving the entire trajectory as was used to solve each segment of the trajectory. Typical practice would be to separate the problem into two or more equal segments depending on the number of orbital transfers.

The full trajectory is shown in Figure 45. The trajectory is described as E-49-E-37-85-E-E. Where the vehicle leaves the Earth and travels to asteroid 49, flyby of Earth, rendezvous at asteroid 37, etc. The orbital elements used for the asteroids listed are described in Table 9.

The full trajectory is challenging to compute and takes over 33 minutes to compute

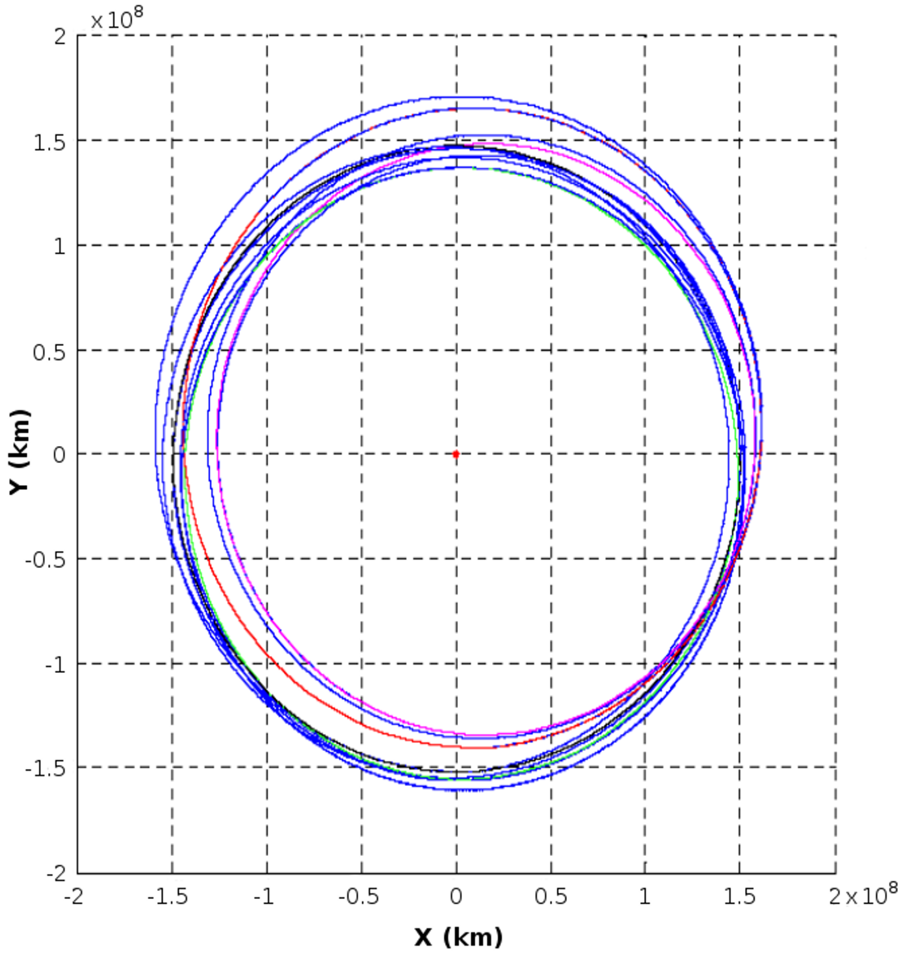


Figure 45: 3rd Global Trajectory Optimization Competition Entry by Georgia Tech.

using the OPTRAIN low thrust trajectory solver developed by Gregory Lantoine.[53] Learning how to separate this trajectory into smaller components that could be used to solve for the original problem is the goal of this section.

3.2.2 Applying Mutual Information to Trajectory Design

The advantages of mutual information will be shown by demonstration on this low-thrust trajectory problem. The decomposition suggested will be compared against: ranking all links equally and attempting to solve the original problem without decomposition. As the original low-thrust problem is normally computationally intensive to solve (e.g. 15-30 minutes per function call), standard practice is to approximate the

Table 9: Orbital Elements for Asteroids used in Problem

No.	**Name**	**Epoch** (MJD)	**a** (AU)	**ecc**	**inc** (deg)	**arg. Peri** (deg)	**Node** (deg)	**M** (deg)
37	2004-QA22	54200	0.9508977	0.12172568	0.57414	28.54873	175.15217	55.2451141
49	2000-SG344	54200	0.9774002	0.06697124	0.11024	274.9223	192.31139	180.3781477
85	2006-BZ147	54200	1.023656	0.09861161	1.40819	95.17529	140.15053	318.8393785

physics of the domain by creating a simplified approximate domain. For example, using ballistic transfers between asteroids rather than low-thrust transfers will decrease computational time to less than a second per ballistic transfer. This approximated ballistic domain provided guesses that were later converged and examined in the true domain.

The dependence information between contributing analyses for this problem (i.e. first visited asteroid, second visited asteroid, etc.) are used to cluster the contributing analyses using mutual information. Links between clusters had their values temporarily frozen while each cluster was run in the true low thrust domain. The sub-solutions found were then reassembled and the total problem solution was refined to provide a measurement of the total run-time when using the decomposition.

The individual legs of the trajectory are shown in Figure 46. Each transfer involves less than two complete revolutions before coming to the next body in the trajectory. The question of how to compute the mutual information of a trajectory transfer was answered by converting porkchop plots into a form that would be meaningful for mutual information.

A porkchop plot describes the change in velocity required to transfer from one orbit to another. The lower areas of the plot describe more favorable windows of opportunity for a transfer, see Figure 47. Two main assumptions are made when

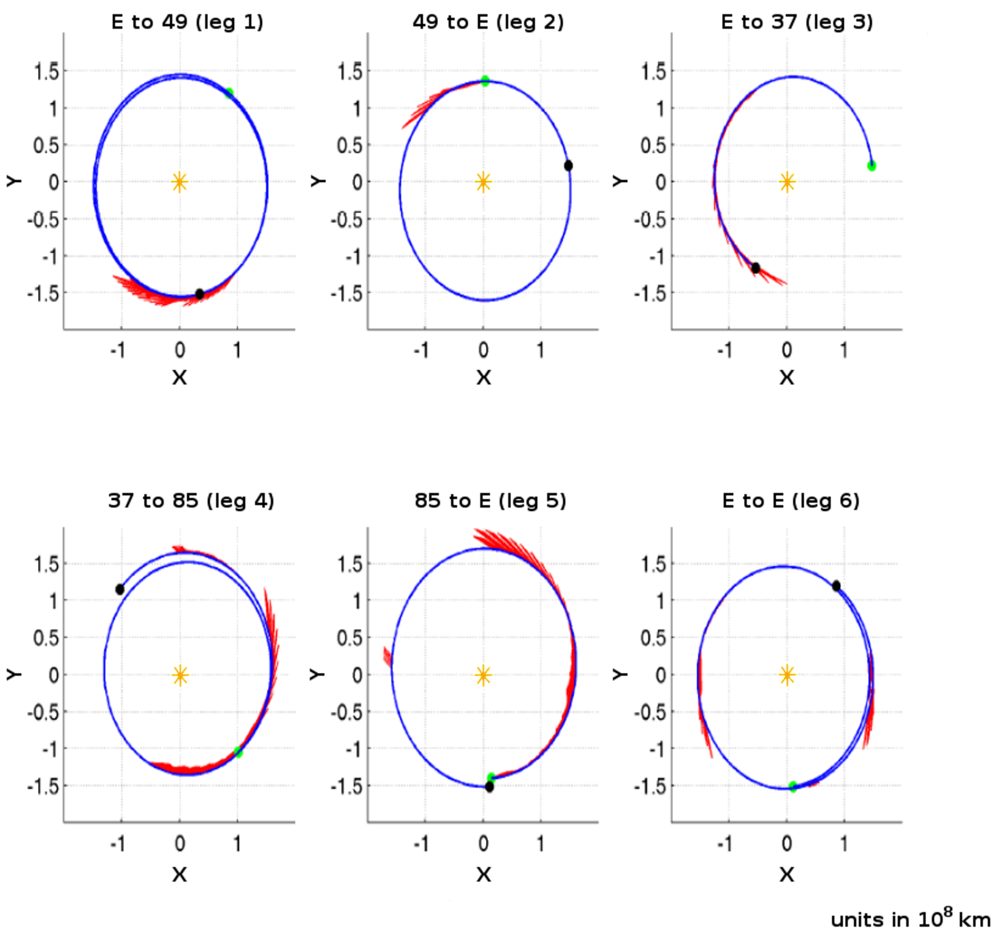


Figure 46: Trajectory Leg Breakdown for the 3rd GTOC.

using these plots as tools for estimating the mutual information between two orbiting bodies. As the plots in this work are based off of ballistic calculations, the assumption is made that the low thrust trajectory is workably approximated by the ballistic calculation. Future could be performed to create plots for this method that are not based on ballistic calculations but on other low thrust approximations (e.g. spiral fitting, etc). The second assumption is that, lacking any knowledge regarding when a transfer will begin, a good estimate for when the transfer begins will be based off of minimizing the required energy to perform the transfer. If one does not know when the transfer leg will start; it is a good estimate to assume that it will likely occur within or near to one of the ”launch windows” of low delta-V.

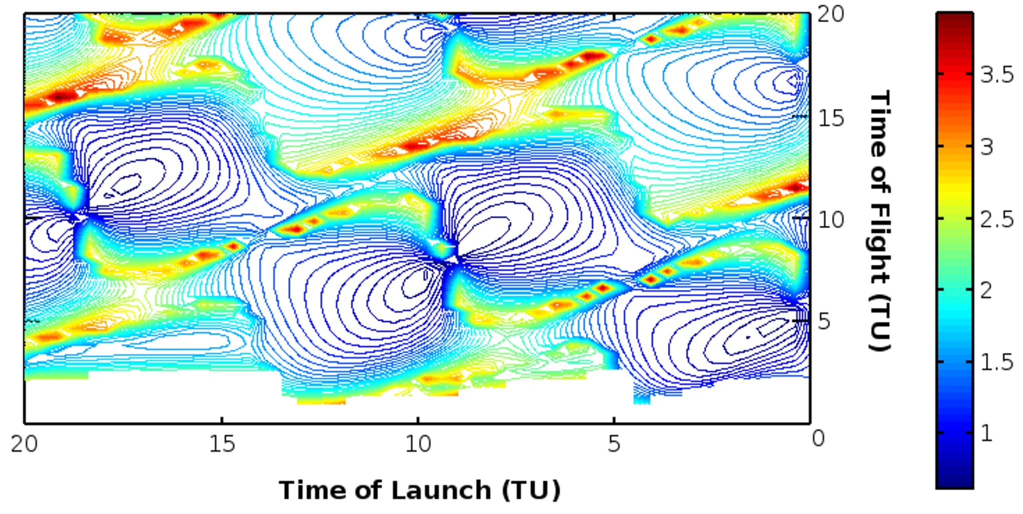


Figure 47: Example of Ballistic PorkChop Plot of Delta-V (DU/TU)

By inverting and normalizing a porkchop plot, a joint probability distribution is formed between the departure date and time of flight. This form stresses those areas of the launch space that would lead to a low delta-V during the transfer. In the distribution, an area is less or more likely in direct relation to the cost of the transfer it describes. Now that the transfer is described generally in the form of a distribution, the mutual information between both random variables can be computed.

This method looks at the trajectory leg as a probability distribution over a range of potential values. The difficulty of a leg can be estimated here through the distribution. Even legs with complicated transfer windows can be compared and ranked against each other.

A transfer that shows insensitivity between launch date and time of flight, such as the plot shown in Figure 49, would have a much lower mutual information than a more complicated chart that only allowed for certain small transfer windows. This follows as it is relatively easy to tack on an extra Earth flyby without changing any transfers from earlier in the trajectory; unless the solution is constrained by total time constraints.

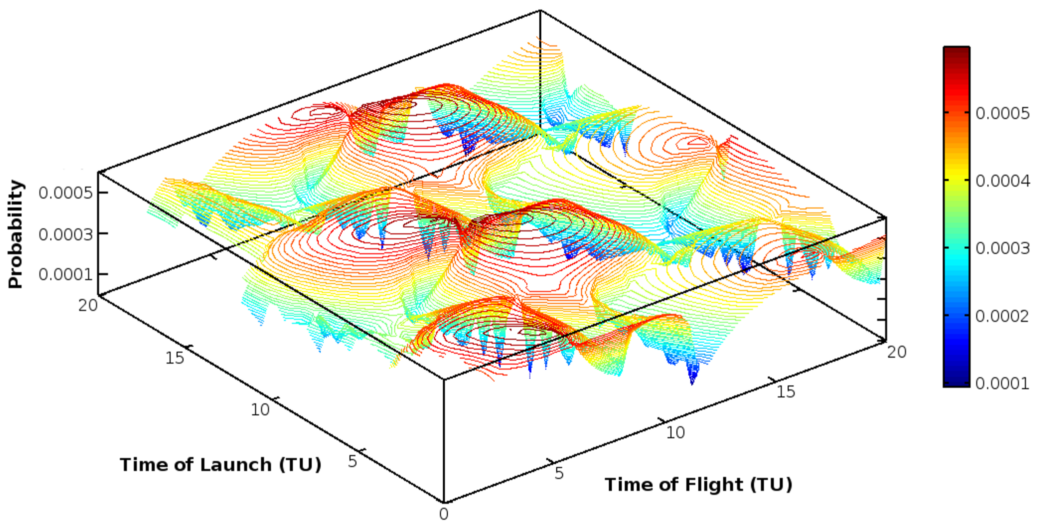


Figure 48: Flipping and Normalizing PorkChop Plot into Probability Distribution

Figure 50 shows the probability distribution formed for an Earth-Earth transfer. The mutual information computed is quite low as there is not much shared information to change the marginal distribution when one random variable becomes known completely. Changing the launch date does not change the marginal distribution for the time of flight variable.

This way of considering transfers that allows a numerical calculation of the dependence between launch date and time of flight. With it, one can numerically estimate the difficulty of trajectory legs which can greatly assist designers who wish guidance while breaking longer trajectories down into more tractable components. Many methods could be developed to add fidelity to the plots used without the need of modifying how the plots are converted to calculate for the mutual information.

3.2.3 Application to a Low-Thrust Problem

Standard practice when confronted by a number of legs is the arbitrarily split the trajectory into smaller pieces that could be run separately; hoping that the consolidated solutions can lead to a final answer for the original problem. The developers of Mystic, a high fidelity low thrust code developed by NASA, suggest that trajectories

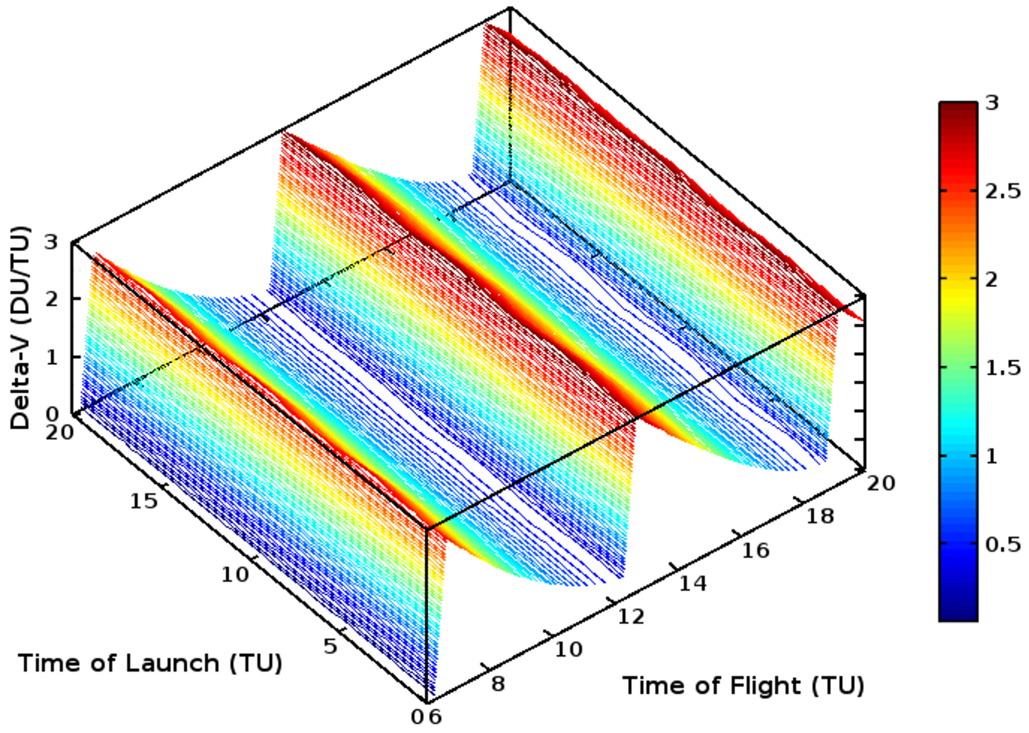


Figure 49: PorkChop Plot for Ballistic Earth-Earth Transfers

with more than 3 flybys separated by interplanetary distances or those with more than 6 close flybys should be optimized in a piece-wise manner.[98] No guidance is provided as to how a trajectory should be best separated.

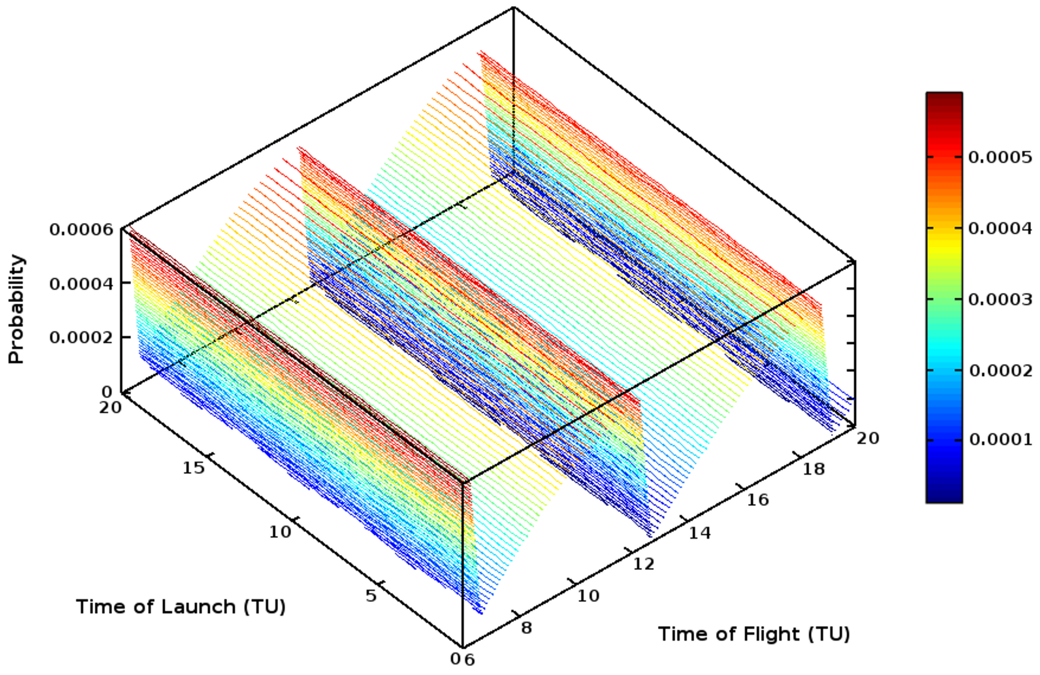


Figure 50: Probability Distribution Formed from Earth-Earth PorkChop Plot

Table 10: Comparison of Cut Performance on Low-Thrust Problem (90% confidence)

Label	Level One RT (s)	+/- (s)	Status at Completion	Level Two RT (s)	+/- (s)	Status at Completion	Total RT (s)	+/- (s)	MI Rank
E-49-E-37-85-E-E	N/A	N/A	N/A	N/A	N/A	Iter. limit; Accuracy Not Achieved; Could not be Improved	2029.7	53.0	-
E-49-E-37 85-E-E	1093.9 145.4	15.6 2.9	Could not be improved Optimal Found	474.6	7.1	Optimal Found	1568.6	22.7	First
E-49-E 37-85-E-E	6.6 291.0	0.1 2.7	Optimal Found Optimal Found	2321.9	167.4	Iter. limit; Optimal Found	2612.9	170.1	Second

The most straight forward method is to separate the trajectory equally into tractable components so this will be used as the example of current acceptable practice. With seven elements, this trajectory could be separated roughly equally in one of two ways. Here the mutual information between the links will be used to aid a designer who would like guidance regarding which is the better decomposition.

Table 10 shows the behavior of the low thrust trajectory problem for both near equal decompositions and for the calculation of the complete trajectory. Mutual information is here calculated or the 37-85 transfer and for the E-37 transfer to see which has a lower mutual information. A cut to the 37-85 connection is suggested by mutual information and performs better than the other equally sized cut. All cuts are run on the same ballistic initial conditions used for the calculation of the complete trajectory. After both segments are solved, their sub-solutions are consolidated into an initial guess for the original problem.

Solving for the original problem is challenging and the program needed to be rerun with its produced updated guess two times. The first time it had reached an iteration limit; its self-produced updated guess was then used to continue the program. The second time it found a near solution but had yet to converge it to the accuracy requested in the input file. The third run was able to converge onto a point that could no longer be improved. Optimality conditions tracked by the software did not call this point an optima but the masses at each asteroid were found to be close to the other solutions found in a piece-wise fashion.

The second row containing E-49-E-37 and 85-E-E took a long time to solve but, as their sub-solutions were more representative of the answers required by the original problem, consolidating both answers was considerably faster. Here mutual information provided a valuable insight that could be leverage into selecting a better decomposition for the problem. The process itself could be automated and the construction of porkchop plots could be easily parallelized.

The last row of Table 10 for the decomposition E-49-E and 37-85-E-E found sub-solutions extremely quickly but these were not compatible towards solving for the original problem. The rework during consolidation increased the total run-time to where less time was spent on the un-separated problem.

In this section, a low thrust trajectory problem was separated into two sub-problems using force-based clustering. These were solved separately and later combined to potentially speed the solution of the original trajectory problem. To rank the importance of each link, a method was created to calculate the mutual information of a trajectory link.

3.3 Summary

This chapter introduced a useful new metric for the evaluation of variable importance. This has wide application to design studies and towards a more complete understanding of variable interaction that often can be non-linear. This metric was compared against correlation and though mutual information on a realistic conceptual design study using the PESST framework for entry system design. A Mars entry trade study was performed to better understand the strengths and weaknesses of mutual information when compared to the standard use of correlation in understanding trade study results.

Though mutual information does not provide directional information on a relationship, it was shown to be able to measure both the linear and non-linear dependence between variables. The metric was able to correctly determine that total propulsion mass was as important as total propulsion system mass (e.g. both are strong indicators for the triggering of the guidance event). Linear correlation was unable to determine the high importance of total propulsion system mass while mutual information was able to correctly determine the most important variables.

This metric provides the ability to better understand the driving variables of a design study. With a better understanding of modeled behavior, synthesis studies can tell engineers more about the systems they model and more quickly expose weaknesses within these model. Here an example showed how mutual information could expose the importance of triggering for the guidance event and the most important driving variables towards meeting the specified objective function.

Force-based clustering was adapted to assist in the decomposition of large problems. Force-based clustering leveraged calculated link importance information to perform a selection between two competing decompositions for a large coupled trajectory problem. A flexible technique for judging the difficulty of trajectory transfers was also presented based on pork-chop plots and mutual information.

The large trajectory problem selected for this example was the Georgia Tech solution for the 3rd Global Optimization Trajectory Competition. Low-thrust trajectories are challenging and time intensive to optimize making the potential to usefully decompose the problem valuable. Standard practice suggests cutting the trajectory into equally sized tractable pieces; this problem had two potential cut-points that would lead to near-equal sub-problems. Force-based clustering was able to successfully determine which cut-point to use and led to a 20% reduction to the total run-time of the trajectory solver.

In the domain of low-thrust problems, designers are often forced to cut long trajectories to meet the computational limitations of the solvers available to them. Force-based clustering and the mutual information metric could provided guidance to where a decomposition should be performed.

CHAPTER IV

VALIDATION OF A NEW OPTIMIZER FOR STATIC DECOMPOSITION

Static decomposition is often used to find a better scheduling of analyses and to uncover closely grouped clusters of analyses that could signify a sub-problem. This type of information can be applied to the structuring of project teams, planning lines of communication between discipline groups, or can reveal physics driving the larger problem by determining problem sub-structure. With a knowledge of sub-structure, there is a potential to speed the execution of larger problems by leveraging newly discovered information. For instance, specialized solvers could be used on sub-problem clusters to discover sub-solutions in the solution to the larger problem.

In static decomposition, the problem is commonly represented as a design structure matrix. This structure is manipulated by an optimizer to find better configurations. The fitness evaluations utilized are performed without run-time data form the contributing analyses. The computational expense of computing the decomposition is preloaded; allowing for a decomposition determination before any contributing analyses are run. No time is required for the run-time calculation of sub-problem structure allowing for the run-time focus of computational resources to be on the execution of analysis codes.

A global optimizer, commonly a Genetic Algorithm, is used with a given fitness function to discover these better performing matrix configurations. GAs are able to search over multimodal solution spaces which contain many local minima that would trap a local gradient based optimizer. These genetically inspired algorithms are also simple to implement and have a long history of use in engineering for searching multimodal domains. The large number of fitness function calls typically required for GAs leads to the common use of inexpensive utility function calls. For the domain of static decomposition, a few common fitness functions are to: limit the number of feedback links or to prefer links that are close to the analyses along the DSM diagonal. When a link is close to the DSM diagonal, the disciplines the link connects are closer to each other on the diagonal. Links to distant analyses are assumed to potentially lead to DSM iteration loops that involve a larger number of analysis block recalculation.

A new global optimizer based on mutual information may provide a number of benefits to suggest adapting it to the field. Two benefits investigated for MIMIC are its ability to converge to high performing solutions using fewer function calls and its potential to more reliably converge to good solutions while using non-optimal settings. Reliability of performance with sub-optimal settings is important as the ideal settings to use on a problem are never known a priori. A preferred method would have its performance be less sensitive to lack of knowledge regarding the settings used.

It would additionally be valuable to know which types of decomposition problems are solved best by the standard GA technique or MIMIC. By testing problems of varying size and complexity, it is desirable to determine where one dominates, where the best can not be decided from current experimental data and where the problem is challenging to both methods. This type of analysis will help direct future method development and provide more information regarding the behavior of the problem domain itself.

4.1 Static Decomposition

This work compares the use of MIMIC to the standard GA for the problem of static decomposition. In this section, this comparison is explained and results from the comparison testing are shown. These results are evaluated as to whether they validate the supposition that a method which models problem structure with probability distributions, MIMIC, can show a marked improvement over a region of the problem domain. The sensitivity of method performance to using near-optimal settings will be examined as a metric for the reliability of the method when applied in a realistic environment.

4.1.1 Approach for Comparing GA and MIMIC

The techniques that follow were required to determine which method converged to a targeted solution fastest. Both methods are stochastic, requiring multiple runs to average their performance on any examined problem. As such, averaged performance is used to judge convergence speed. The experimental plan is also designed to determine which method is more robust regarding sub-optimal method settings. Optimizer performance is sensitive to the settings used to solve a problem but this sensitivity will differ between methods and between problems. The sweep of problems examined will aid in determining averaged sensitivity characteristics for GAs and MIMIC.

The optimization of a function can be challenging for Genetic Algorithms. GAs assume a building block hypothesis where sub-solutions can be used as building blocks to larger solutions.[36] Most crossover techniques used assume that building blocks share a close proximity on the candidate string that is often true. Additionally, it has been observed in practice and literature that method settings are not reliably transferable between similar problems.[102] The method that is more robust to non-optimum settings or the method with problem settings that are easily predictable is desired for the challenge of static decomposition.

To measure the sensitivity of methods to an imperfect knowledge of their settings one needs to compute cases over a range of different settings. The best settings to use in that range for the problem are found and examined to determine the performance when the settings are near but not at this best performing point.

4.1.1.1 Examined Domain of Optimizer Settings

The ranges utilized for each of the methods are outlined in this section. These ranges are over three settings for each of the methods. Population size, crossover percentage and mutation rate are modified for Genetic Algorithms. Meanwhile the candidate list size, percentage used for modeling and a noise parameter are examined for MIMIC. The best settings for the optimizer, within the stated setting domain, was determined by the exploration of 45 points in the setting domain selected by a mixed Latin Hypercube and 2-level Full Factorial Design of Experiments. The best performing point was taken as the best set of settings for this region. To measure the near-best performance of the optimizers, a cube centered about the best found point was used to calculate performance when using near-best settings.

Table 11: Setting Domain Searched for Genetic Algorithms.

Method Setting	**Low**	**Mid**	**High**
Population Size	100	200	300
Crossover Likelihood	70%	85%	100%
Mutation Likelihood	5%	10%	15%

When crossing two different arrangements of analyses, each analysis can only appear once in the child. This spurred the development of several different crossover types that maintain this restriction. Two crossover types commonly used for arrangement problems, order-based crossover and cycle-based crossover were examined in this study. Cycle crossover consistently performed better than order-based crossover in preliminary tests over the range stated in Table 11. As a result, cycle crossover was used as the default approach for this investigation.

Preliminary testing over the test problems was used to generate conservative bounds for the setting regions used. The crossover likelihood in Table 11 was anchored on one side by the value 100 as a high crossover rate is common for Genetic Algorithms. A range of population sizes from 100-300 was selected for Genetic Algorithms. Later plotting of all results over this domain of settings showed that the population size setting was not a main driver of performance for this problem range. The likelihoods selected for both the crossover and mutation settings were found to be far more important and will be examined later.

For MIMIC, the candidate list size is similar in function to the population size in genetic algorithms. The name is kept different to avoid the mistake of thinking that MIMIC relies on genetic operations or is another type of GA. The main difference between them is that the size of the candidate list determines the maximum possible sample size that could be used to calculate probability tables in MIMIC. The Percentage of List used for Model setting in Table 12 determines what percentage of the total potential sample size is used to calculate these probability models.

Table 12: Setting Domain Searched for MIMIC.

Method Setting	**Low**	**Mid**	**High**
Candidate List Size	100	1550	3000
Percentage of List used for Model	35%	50%	65%
Probability Noise Percentage	1%	6%	11%

There is always the risk that the sampled candidates do not possess the information needed to converge onto better performing solutions. To help avoid over-fitting probabilities from the use of a finite sample size, a noise setting is added. The noise parameter determines the percentage of the final distribution that comes from a uniform probability over all currently available options. For the stored marginal and conditional probabilities used by MIMIC, a composite probability is calculated as shown in Equation 21. 100% minus the noise percentage comes from the learned distribution.

$$prob_{\text{composite}} = (1 - \text{noise})\, prob_{\text{from model}} + (\text{noise})\, prob_{\text{uniform}} \tag{21}$$

The work of Bonet et al.[8] set the percentage of candidates used for model building to 50%. Therefore a range was specified of 35%-65%.

The variance in performance about the best performing settings can be measured by examining a cube of points centered about the best performing set of settings. By using the same number of settings for each method, the dimension of the cube formed about the best performing point is the same for both methods (i.e. a 3D cube). This aids in having an apples-to-apples comparison when examining the range of performance within this cube. Points were taken outside of the regions in Table 11 and Table 12 in case the best performing point was found to rest along one of the region boundaries. In that case, the best performing point in the reported region is used to center the cube.

Table 13: Design of Experiments Breakdown

Category	**Number of Points**
2-Level Full Factorial	8
32 point Latin Hypercube	32
Randomly Generated Points	5
Total	45

Table 13 describes the Design of Experiments (DOE) that was used to examine the range of settings in Table 11 and Table 12. The DOE is based initially on a 2-level full factorial design that places its experimental points along the edges of the design space. Sampling from the interior is performed by using a 32-point Latin Hypercube. Five randomly scattered samples from the domain are recorded for validating any potential model created of the domain behavior. This amounts to reserving 12.5% of the points generated for model validation.

As stated earlier, the stochastic nature of the methods require that any combination of settings should be independently tested multiple times to record their averaged performance. Every one of the 45 setting combinations in the DOE from Table 13 are independently reinitialized and rerun ten times to record both the mean and mean confidence bounds. This amounts to $45 \times 10 = 450$ individual cases that must be separately run to determine the settings that produce the best discovered averaged performance for one method and one problem. This was repeated for each method and problem considered.

The forty five problems investigated for this study were randomly generated. A target fitness value for judging convergence was calculated by attempting to solve for each problem by the GA ten times. The best found fitness value was used as a target for both optimization methods. A combination of settings is considered as converged if the problem target value was met or surpassed before reaching an upper function call limit of 2 million calls that is used to bound the computation. The combination of settings that leads to the lowest number of unconverged cases and then to the lowest run-time was used to select the best performing point. The number of unconverged cases, number of function calls used, and run-time was compared between the two optimizers for both speed and reliability are important attributes for an optimization method.

4.1.1.2 Test Requirements for Pseudo-Random Number Generation

Pseudo-random number generators are used to generate simulated random values that are imitations of independent and identically distributed (iid) values.[55] These are deterministic algorithms that generate random numbers whose behavior is meant to be very hard to statistically distinguish from truly random numbers. In order to run several repeated experiments, high quality pseudo-random numbers are required to ensure independent results. Analysis results could not be considered as iid without high quality random numbers. Additionally, confidence intervals computed on this averaged behavior would be much harder to compute and defend without iid experimental results.

Programmatic methods to simulate randomness will begin to repeat after a known number of generated pseudo-random numbers. Methods each have different performance characteristics when tested for statistical randomness, for instance the default rand function in MATLAB and C have historically been too poorly random for use in Monte Carlo analysis.[43] Researchers normally switch to other pseudo-random number generators that are better able to simulate random properties. An example generator often used for Monte Carlo analysis is the Mersenne Twister algorithm.[61]

The Keep It Simple Stupid (KISS) algorithm from Marsaglia[43] is used in this study to generate pseudo-random numbers. It generates high quality random numbers and is one of the simplest methods to pass the DieHard[60] series of tests for random number generators. It is composed of three different types of pseudo-random number generation methods where the weaknesses of one method type over a region are negated by the other two method types over the same region. A stronger combined method results that is still straight forward to implement.[43]

For this study, one KISS generator is initialized with four seeds from highly entropic data normally stored on a Linux computer within /dev/random. This file stores random information from when drives are read, files accessed, etc. Programs take longer to generate numbers from this source but they are of very high quality.[21, 43] This initial KISS generator, initialized from /dev/random, is used to generate seeds for all other KISS generators used.

One independent KISS generator is used for each candidate solution in the population for each method. This is to better ensure high quality randomness and, by providing one generator for each candidate, no read/write conflicts occur during the concurrent modification of the currently used seeds. This means that no software locks are required to control access to the memory used by the seeds allowing for simpler and potentially faster code at the cost of a larger memory footprint.

4.1.1.3 Problem Set Utilized for Study

Characteristics of the problems selected for this comparison study are examined here. For the study, it is desirable to examine problems over a range of size and complexity. Three difficulty levels (small, medium, large) were applied to size and to the level of coupling present in the problem. The numerical values associated with these levels are specified in Table 14 for a total of nine problem characteristic combinations.

Table 14: Problem Testbed Selected for Comparison.

Attribute	**Small**	**Medium**	**Large**	
Problem Size	15	25	50	3 steps
Links Active	5%	10%	20%	3 steps
Total Configurations				9
Random Problems Generated per Configuration				5
Total Randomly Generated Problems				45

Graphically, an example problem showing each of these configurations is shown in Figure 51. The links active metric describes the percentage of potential link locations in the DSM that are active. The blue boxes specify the locations of contributing analyses. Tying these DSM sizes to the literature, the larger problem presented for a General Motors Engine from Chapter 1 in Figure 3(b) is composed of 24 contributing analyses while a provided example for an automobile design process had 15 analyses, Figure 6(b).[15, 62] As such, 50 contributing analyses was selected to represent a large problem for this work.

When creating the problems, the link placement for each problem was randomly selected using a KISS generator with the only restriction being the specified problem size and the total number of links present. This method was used to prevent the

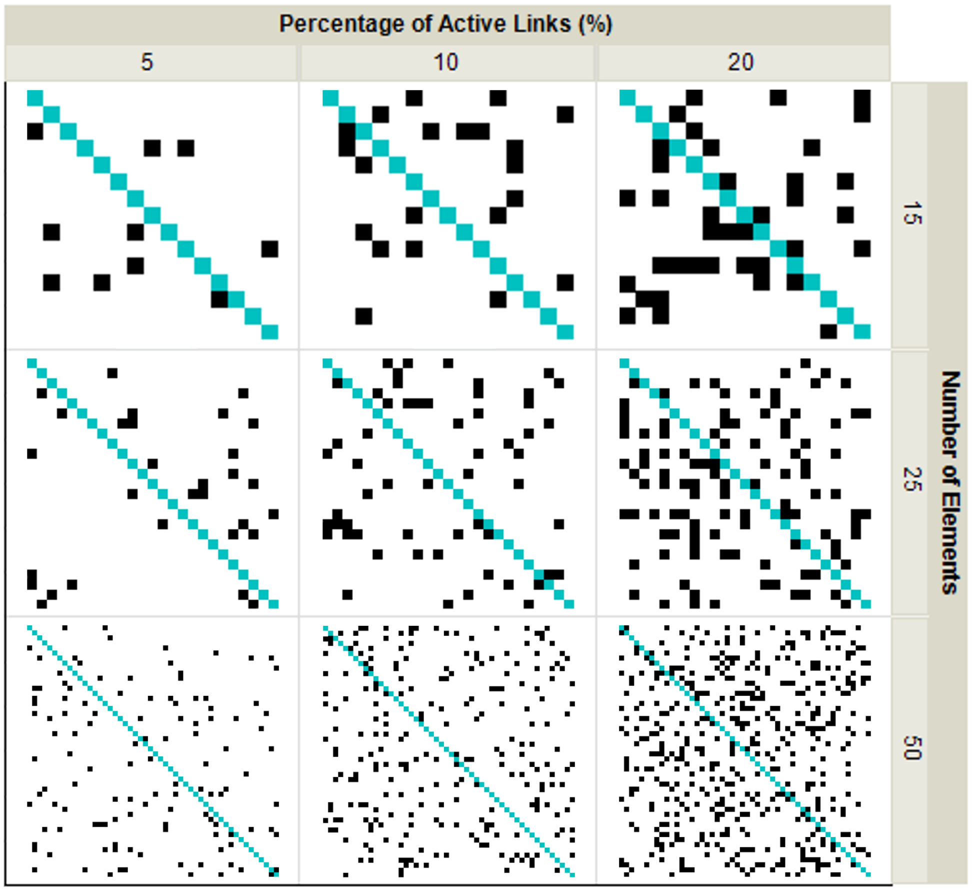


Figure 51: Example Initial Configuration for Different Problem Difficulties Examined.

cherry-picking of problems that might give the impression of bias to the study. This problem set is sampled from the complete set of problems of that size and complexity. Five different randomly created problems are created for each combination of size and complexity characteristics to form a set of 45 problems. Repetitions of each size-complexity configuration were used to average comparison results for each problem configuration.

Both methods are provided with a target utility that was determined for each problem. Once the target value is found the method stops and performance metrics are taken. This aims to provide a fair comparison between the methods that would not exist if one or the other were allowed to converge to a weaker solution. The performance metrics are all with reference to when the method obtained a utility value that is at least as good as the specified target value. To find the target utility for a randomly created problem, all of the problems were initially solved multiple times by a genetic algorithm and the best discovered solution used as the targeted utility. Once the target value or a better utility is found, the method is considered as converged

A maximum number of function calls has been set at 2 million. This is to avoid a potential infinite loop if a method becomes trapped within a poor local optima. As each method uses a different number of function calls for each iteration, a function call limit was a more fair way of capping the computation between the methods. If either method reaches this maximum number of function calls it is recorded as unconverged. It is valuable to compare the number of unconverged cases to see the propensity of a method to become stuck within lower performing minima, within the setting domain explored for this study.

4.1.2 Computational Framework for Comparison Testing

This section will describe the software framework developed for the comparison of these two methods and explain how each method was written to utilize parallel computation. Given the number of cases examined, parallel computation provides the computational power to perform a detailed comparison between these methods for Static Decomposition. This should be useful for others who wish to create parallel implementations of these algorithms.

4.1.2.1 Need for Parallelization

With 45 setting combinations, each run ten times for statistical measurements and for two methods, a total of 900 cases need to be run for each of the 45 examined problems. Each case does not rely on the calculation of any other case and could be run concurrently. These cases would take years if calculated in a serial manner which is one reason that this type of comparison is rarely seen in the literature and is completely novel for the analysis of the MIMIC algorithm. Advances in both hardware and programming interfaces have made massively parallel programming greatly more available to researchers. Inexpensive and highly parallized hardware is starting to become more available though recent developments in graphical programming units. These developments are leading to drastic reductions in computational time for studies that can utilize this available hardware. NVIDIA graphics hardware was utilized for this research with the Compute Unified Device Architecture (CUDA).

4.1.2.2 Processing using CUDA

CUDA is a parallel architecture for utilizing the parallel processing abilities of NVIDIA graphics cards. In practice, it serves as an extension to the C language which enables the specification of parallel code that will be run using multiple threads on the graphics card. The first release of the CUDA application programming interface in 2007 greatly aided programming to NVIDIA hardware. The CUDA 2.0 framework was utilized for this study.

The processing capability of graphics cards has out paced CPUs over that last several years (Figure 52). Though CPU cores are more flexible in terms of what types of processes they are optimized to run, GPU cores have been specialized for numerical calculation which makes them well suited for many scientific applications. Historically this specialization was driven by the large parallel numerical calculations typically required during the rendering of graphics for progressively more intricate video games.

Current hardware limitations on the GPU guide algorithm development on the cards. While numerical calculation is rapid on the GPU cores, there is a potential bottleneck in the time it takes to transfer data to/from the graphics card. Data traffic

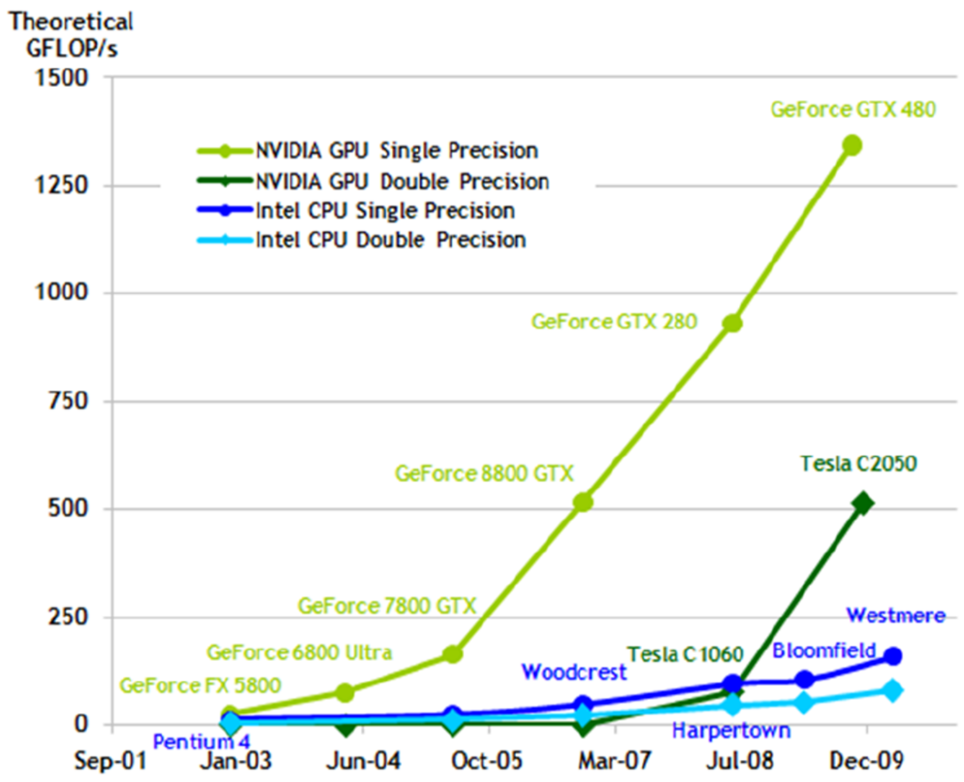


Figure 52: Historical Performance Comparison between CPU and GPU.

between the card and hosting computer could strongly limit the benefits of parallelizing the software. It is much slower to transfer data than to execute instructions on the card. This has guided the development of the parallel implementations of genetic algorithms and MIMIC by the intention to minimize data traffic to the card.

Both genetic algorithms and MIMIC are made to run start-to-finish on the graphics card, minimizing the transfer of data between the host system and card. All required data is transferred to the card during problem initialization and then the method, from that point, completely executes on the card only transferring back the final answer or to have a different set of threads initialized. If control needs to be passed back to the CPU, for it to call another GPU routine, only pointers to data structures already on the graphics card are passed between the GPU and CPU. This allows the next GPU routine to continue execution on the same data while only passing a few bytes between the GPU and CPU. By utilizing parallel hardware and a parallel implementation for these algorithms, a comparison is enabled that is unusual given the massive computational effort that would normally be required.

4.1.2.3 Parallel Implementation of Genetic Algorithm

For this study, several genetic operators have been implemented to run on an NVIDIA video card. Reproduction has been implemented to use tournament selection to pass on individuals that win the pair-wise fitness comparison. Here elitism is used to always preserve the best performing candidate between generations. Two crossover methods suggested from the literature for scheduling problems were examined preliminarily: order-based and cycle crossover. Cycle crossover was selected as the method to use over the range of settings in Table 11 as it consistently performed better over the region and problems examined. Finally, mutation has been implemented by the random selection of two DSM analyses and swapping their locations in the DSM.

GPU bandwidth limitations have been minimized in the algorithm by reserving the memory required by the algorithm during initialization. The entire GA is then run on the graphics card until it converges or hits the maximum fitness call limit. Genetic algorithms are well suited for parallel computation. Each candidate is handled by a separate computational thread that performs each genetic operation on the candidate. Fitness evaluations, tournament selection and mutation are trivially parallelizable for an individual in the population. The crossover operation was parallelized by holding two populations in memory; the current population and the population for the next iteration. This allowed multiple threads to modify individuals during the crossover operation without the need for memory locking as the original individuals were preserved in the current population for reading by all threads.

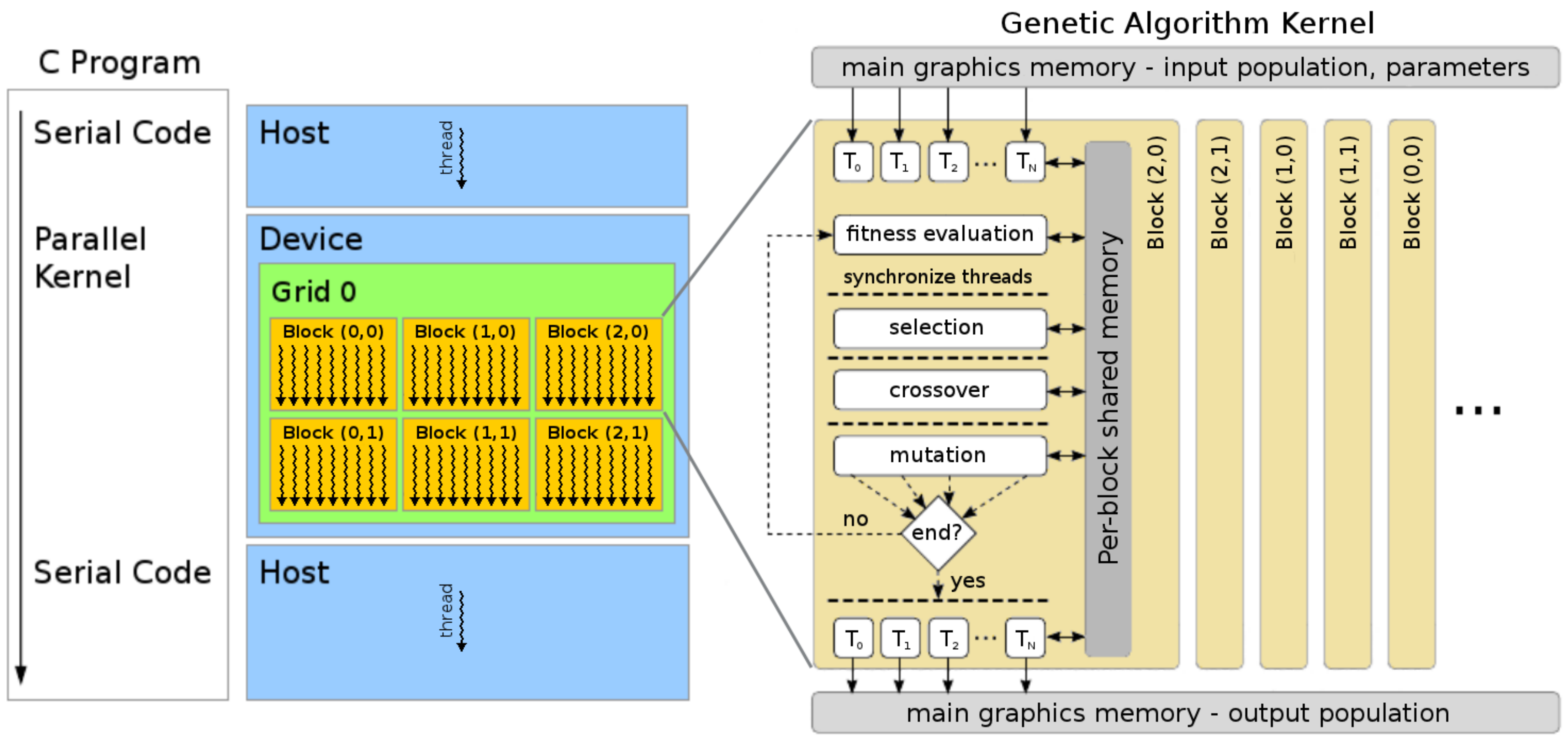


Figure 53: Genetic Algorithm Kernel for Video Card.

Figure 53 shows how the GA could be separated out to run through several threads on the graphics card. Each individual in the population is handled by one thread throughout the running of the method. Every block of threads, on the card, represent a separate case that can be run concurrently on the GPU. Every thread has access to its own KISS random number generator.

4.1.2.4 Parallel Implementation of MIMIC

The implementation of the MIMIC algorithm also followed the technique of keeping the list of candidate solutions in the memory space of the video card during the entire run of the method. This was done again to limit the interaction between the card and main memory. The challenge with MIMIC parallel implementation came when implementing two different steps, model building and fitness function evaluation, in the algorithm these were best written using a different number of threads.

It was most straight forward, when model building, to use one thread for each of N vertices in the initial fully connected graph, by Table 14 N is between 25 and 50 inputs (i.e. vertices). Each thread needs to compute the mutual information between itself and the $N-1$ other vertices. As the links are undirected, only half of the full $N \times (N-1)$ links need to be calculated during the creation of the model. Additionally, the mutual information calculation only requires read access to the list of candidate solutions to be modeled. This allows for multiple threads to use the candidate solutions concurrently without the run-time penalty of data locking. A parallel minimum spanning tree algorithm has been adapted to create a parallel maximally connected tree algorithm. The parallel minimum spanning tree algorithm that was modified is described in the work of Harish et al.[35].

Fitness function evaluation is easiest to implement when there is one thread for every candidate solution so while the number of vertices N might be 50, the number of candidate solutions could be 300. This suggested a need to change the number of

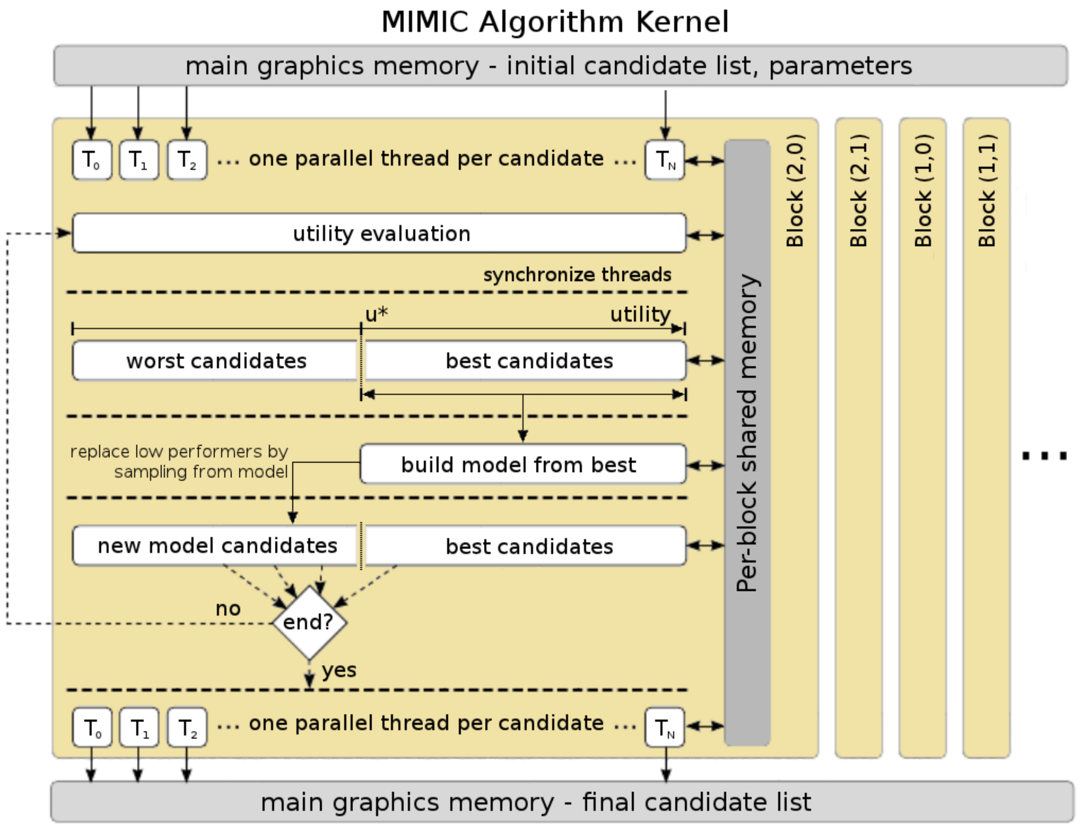


Figure 54: MIMIC Kernel for Video Card.

threads used for each action. To use different thread numbers, two approaches were considered. The first was to specify the use of the largest number of threads that would be required by the method steps. This was rejected as it would inefficiently leave many threads unused during execution. The second and more efficient solution was to temporarily shift control to the CPU where it could set a different number of threads to perform the next action. The possible data transfer penalties were mitigated by only passing pointers to data that already existed on the graphics card. In Figure 54, every dashed line to signify synchronization also passed control back to the CPU where it could call the CUDA kernel for the next step in the method with an appropriate number of threads.

The parallel implementation for new candidate generation, from the formed model,

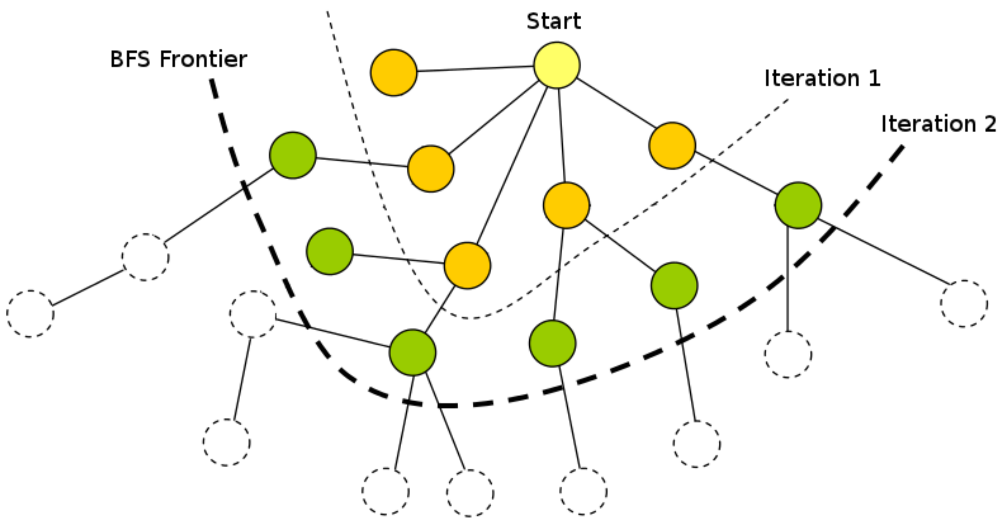


Figure 55: breadth-First Search of Tree Model, image modified from [35]

utilizes a breadth-first search through the formed tree model. Here one thread is used for every node in the tree model where nodes that have the required data to calculate will compute an input value for every new candidate. At the start, only one node, start, has enough data to calculate its value. After the start node has been computed, every one of its children will have the information required to run and record their sections of the new candidate solutions, see Figure 55. The usual technique used by MIMIC, when using one thread, is to use a depth-first traversal of the tree. A breadth-first traversal is suggested here as an improved means of generating new candidates when using multiple threads. Instead of the depth-first traversal which takes N iterations, the number of breadth-first iterations is only N for the worst case tree where the start node is at the end of a straight chain of nodes. Normally, breadth-first will be much faster as more than one node could be completed per iteration.

Comparing the implementation difficulty to genetic algorithms, MIMIC is more challenging to implement but not greatly so. Probability is needed for both though with MIMIC probability tables need to be kept. A maximum spanning tree algorithm needs to be found or implemented. Several implementations for finding the minimum spanning tree are available and only small changes are required to make such an algorithm solve instead for the maximum spanning tree. The greatest challenge for MIMIC relative to GAs is simply in understanding the steps required for implementation of the algorithm and the reasoning behind them.

4.1.3 Evaluation and Validation of Static Comparison

This study was performed on the Keeneland computer cluster which is managed by the Georgia Institute of Technology, Oak Ridge National Lab, the University of Tennessee-Knoxville, and the National Institute for Computational Sciences; funded in large part by the National Science Foundation. The cluster has 120 computer nodes that make available 240 Intel Westmore hex-core CPUs and 360 NVIDIA 6GB Fermi GPUs. All stated run-times are on these NVIDIA Fermi GPUs.

This work far surpasses the current practice of only using a few problems for testing that are often not generated randomly. This work was enabled by months of computational time on the Keeneland cluster. Quantitative problem behavior of a higher fidelity could be obtained by a larger selection of problem sizes, complexity measurements, and a larger number of randomly generated problems for each configuration. This exhaustive analysis is beyond the current hardware capability of our systems. The selection of problems and configurations used for this work provide valuable trend information to aid in selecting a method for static decomposition. This work provides the highest fidelity comparison between Genetic Algorithms and MIMIC currently in the literature.

GA and MIMIC performance, over the range of settings investigated, will include a comparison of function calls required to meet a target utility value, the percentage of cases that did not converge by 2 million function calls, and the total averaged run-time taken by the methods during the execution of the problems.

A search over a setting domain was used to discover a best-found set of settings for each method. Results for each performance metric are provided for this best set of settings. An additional analysis that is rare for the literature is provided where the performance of both methods are also provided for a margin from this best-found set of settings. This will provided the user with knowledge of how either method fairs when near-best settings are utilized. As the knowledge of the best settings are not available in practice, near-best performance is a more realistic comparison of method performance.

4.1.3.1 Results using Best and Near-Best Settings

The 45 problems in Table 15 were randomly created to have the one of nine configurations of element number and link complexity. This provided for five different problems for each explored configuration, see Table 14. These five problems were used to develop averaged performance estimates for each of the nine problem types examined.

Table 15: Detailed Static Comparison Results Per Problem (Best Method Settings)

					90% Confidence Interval on Average				90% Confidence Interval on Average			
ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
46	15	5	0	0	140.7	20.66	155	27.92	10.58	1.907	1.798	0.3239
47	15	5	0	0	156	23.03	215	43.48	13.03	2.381	2.08	0.4207
48	15	5	0	0	261	42.68	370	60.97	33.63	6.214	2.823	0.4651
49	15	5	0	0	278.5	46.12	355	81.36	35.87	6.656	2.786	0.6385
50	15	5	0	0	278.5	45.35	365	69.87	37.05	6.762	2.79	0.534
0	25	5	0	0	5.05E+03	404.3	1.20E+03	217.6	1.18E+02	9.446	1.16E+01	2.103
9	25	5	0	0	5.94E+03	427.7	1.54E+03	138.8	1.26E+02	8.635	1.22E+01	1.097
18	25	5	0	0	9.06E+03	1419	3.93E+03	385.1	2.02E+02	30.57	1.81E+01	1.773
27	25	5	0	30	3.19E+04	16130	6.27E+05	339900	7.41E+02	370.8	2.01E+03	1088
36	25	5	0	0	3.80E+03	515	9.75E+02	94.63	1.07E+02	15.79	1.04E+01	1.004
1	50	5	0	0	5.95E+04	5229	1.26E+04	1093	1.20E+04	1052	1.53E+02	13.24
10	50	5	0	45	4.55E+05	76720	1.24E+06	321300	2.37E+04	3986	1.02E+04	2645
19	50	5	20	50	8.37E+05	348200	1.24E+06	298400	6.74E+04	28040	3.33E+04	7987
28	50	5	20	55	1.08E+06	330300	1.24E+06	325300	1.20E+05	36570	1.40E+04	3687

Table 15: Detailed Static Comparison Results Per Problem (Best Method Settings)

					90% Confidence Interval on Average				90% Confidence Interval on Average			
ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
37	50	5	40	55	1.18E+06	406200	1.42E+06	290000	1.97E+05	68130	7.14E+04	14600
51	15	10	0	0	3085	351.2	6270	2030	225.2	25.8	15.26	4.941
52	15	10	0	0	3614	1181	3825	430.9	161.6	52.09	10.24	1.153
53	15	10	0	0	964.5	137.3	1150	176.2	85.15	11.75	5.581	0.8549
54	15	10	0	0	2945	1485	3538	740.9	308.6	155.2	9.956	2.085
55	15	10	0	20	9728	2826	4.26E+05	2.98E+05	436.6	126.2	1108	774
3	25	10	0	15	4.13E+04	5246	4.30E+05	253300	1.11E+03	140.4	1.42E+03	836.3
12	25	10	0	0	1.30E+04	945.7	4.94E+03	1097	2.99E+02	21.3	3.06E+01	6.802
21	25	10	0	25	2.94E+04	3225	5.82E+05	313100	2.10E+03	230.1	2.08E+03	1120
30	25	10	0	25	4.75E+04	4594	6.85E+05	299400	1.38E+03	134.3	1.18E+04	5145
39	25	10	0	15	2.23E+04	3567	3.31E+05	265200	5.80E+02	92.42	1.17E+03	936
4	50	10	0	55	5.77E+05	203200	1.36E+06	297100	4.15E+04	14640	3.13E+04	6858
13	50	10	0	20	3.60E+05	43400	9.59E+05	277500	2.81E+04	3385	9.96E+03	2883
22	50	10	0	0	1.80E+05	29210	6.12E+04	21400	5.86E+04	9529	7.14E+02	249.6
31	50	10	0	55	3.35E+05	37080	1.26E+06	317700	2.41E+04	2668	3.78E+04	9549

Table 15: Detailed Static Comparison Results Per Problem (Best Method Settings)

					90% Confidence Interval on Average				90% Confidence Interval on Average			
ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
40	50	10	20	55	8.28E+05	363400	1.43E+06	264800	4.01E+04	17580	7.49E+04	13910
56	15	20	0	10	1.17E+04	1799	2.49E+05	2.23E+05	597.9	91.51	518.8	463.3
57	15	20	0	10	1.08E+04	1213	2.53E+05	2.25E+05	678.8	76.77	537.5	477.1
58	15	20	0	0	1.21E+04	2720	7.55E+04	8.76E+04	800.9	182.3	517.7	601.2
59	15	20	0	0	4835	946.5	4066	585	220.8	42.78	12.99	1.869
60	15	20	0	0	7088	692.1	3.62E+04	2.65E+04	264.8	25.22	78.49	57.61
6	25	20	0	10	5.99E+04	6232	4.00E+05	234600	1.33E+03	137.6	1.63E+03	957
15	25	20	0	0	5.60E+04	6839	1.89E+05	78260	1.70E+03	208.5	7.56E+02	313.7
24	25	20	0	5	2.98E+04	3328	3.06E+05	193400	7.22E+02	81.08	1.53E+03	963.2
33	25	20	0	0	3.38E+04	2903	3.85E+04	23340	1.01E+03	86.81	1.78E+02	107.7
42	25	20	0	0	3.69E+04	3134	1.70E+05	144200	1.10E+03	94.68	7.36E+02	624.3
7	50	20	0	0	2.14E+05	31450	1.22E+05	31330	1.66E+04	2433	2.01E+03	513.9
16	50	20	0	60	1.23E+06	172600	1.43E+06	279000	1.89E+05	26610	2.32E+04	4509
25	50	20	0	20	3.13E+05	38910	8.75E+05	265600	1.28E+04	1585	1.34E+04	4051
34	50	20	0	75	1.26E+06	100500	1.72E+06	202400	1.46E+05	11650	2.64E+04	3115

Table 15: Detailed Static Comparison Results Per Problem (Best Method Settings)

					90% Confidence Interval on Average				90% Confidence Interval on Average			
ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
43	50	20	0	35	6.78E+05	172400	1.14E+06	299400	5.07E+04	12900	1.54E+04	4030

The method comparison shown in Table 15 used the best settings discovered for each method. The best found settings used to generate these results are provided in Table 36 in Appendix B. These results show both method's performance over a range of different problems. This was done to discover the areas of the problem domain that are better solved by either method. In addition to the best settings found, a cube of points in the setting domain were computed around these best performing points to determine the sensitivity of both methods to near-best settings. The points of the cube were set at +/-20% from the best found settings specified in Table 36.

The best set of settings are never known in practice before a problem is executed. The comparison over near-best settings is aimed to provide a more realistic estimate to the performance of either method. One method is considered as superior when the confidence interval of the difference does not cross zero. If the confidence interval of the difference crosses zero than the experimental results cannot rule out the null hypothesis that their behavior is equivalent for the problem investigated. If the null hypothesis can be ruled out, by not crossing zero, then the method with a better average performance is considered as superior. Unless otherwise stated, the confidence bounds on the mean is 90%. Comparison results for different confidence intervals can be determined from Table 34 when using the best settings or from Table 35 for near-best performance behavior. These Tables are available in Appendix B.

4.1.3.2 Function Call Comparison Between Optimizers

Function calls are used to determine how well the methods utilize the data they obtain through examining the problem. As the cost of function calls increase, this performance metric becomes one of the most important to the run-time of the method. Each of the nine problem configurations examined is shown in Figure 56 as a circle. Each of the circles is separated into five segments that represent the five problems examined with those problem characteristics. The superior method is noted by the color of the segment. 'Unknown' is used if there is insufficient statistical information to differentiate between the methods to a 90% confidence on the mean.

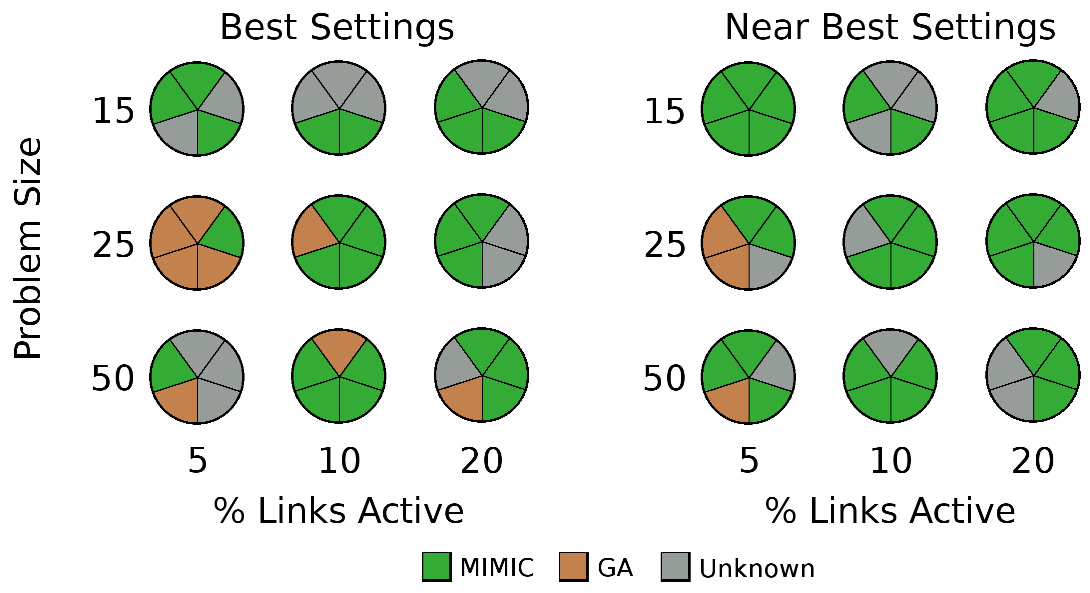


Figure 56: Function Call Performance over Problem Set (Best and Near Best Settings)

When using the best settings found for function call performance, Figure 56, MIMIC is superior on the four configurations with medium to high size and medium to high complexity. When assuming knowledge of the best performing method settings, the genetic algorithm is superior on the loosely coupled version of the medium sized problem configuration.

The near best settings used in Figure 56 show the averaged function call performance when one uses near-best settings within +/-20% of the best settings from Table 36. This estimate assumes that a user with some idea of which settings to use would fall within 20% of the best settings that would be found through a thorough search. The differences between the best and near-best performance is explained by both a larger sensitivity of GAs to the settings used and the slightly better performance of MIMIC on several of the problems. Though the confidence bounds are overlapped for several of the individual problems, MIMIC consistently has the bound that ranges lower. Now the advantage GAs experienced on loosely coupled medium sized problems becomes inconclusive to a 90% confidence when near-best settings are used.

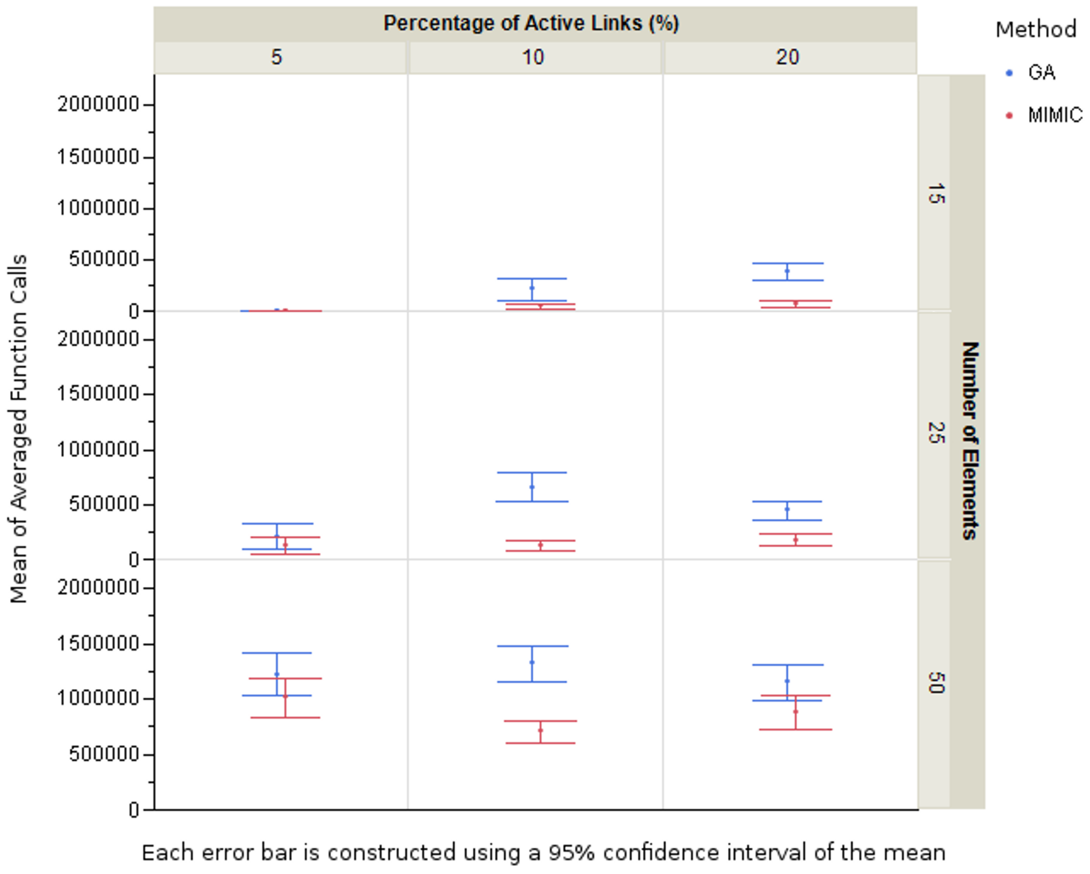


Figure 57: Mean Function Call Performance over Problem Set (Near Best Results included in Average)

The averaged function call performance using near best settings is shown in Figure 57. This is equivalent to the averaging of each of five problems from Figure 56 and allows the reader to see the confidence bounds for each method over the range of problem types. MIMIC, the method shown in red, is consistently lower and has either tighter or equivalent confidence bounds when near best results are included in the performance average.

MIMIC was shown to better utilize the information gained from each function call.

This will become useful for problems with more expensive function calls assuming that the convergence behavior seen here is roughly indicative of the behavior on a different utility function.

The number of function calls used by the GA for the 50 element problem though was almost always greater than MIMIC. This comes from the fact that roughly 50-60% of the cases run by the GA at the large problem size failed to converge. Even if a method were to take more function calls, it would be allowable if it had a better chance of reliably converging onto the answer. The higher expense can be forgiven for the sake of higher reliability. For these problems, MIMIC could both more reliably converge and have a lower total function call count.

4.1.3.3 Convergence Comparison Between Optimizers

To measure the reliability of both methods, the percentage of cases that failed to converge within the two million function call limit was tracked. When the confidence bounds on the difference between both method distributions crosses zero, the determination of a winner is stated as 'Unknown' as it was above.

MIMIC surpasses GAs in convergence reliability when using the best settings found in Figure 58. The near best settings create some overlap in the computed confidence intervals but MIMIC remains as either the superior or equivalent choice.

MIMIC in the averaged near-best results, Figure 59, are low and tight for the 25 element problems while GAs had less reliable behavior. The GA had roughly 30% of problems fail to converge for 25 element problems of medium complexity while MIMIC had a low percentage of unconverged cases for all 25 element problems tested. For large sized problems, GAs were challenged as 40-60% failed to converge compared to a more reliable performance from MIMIC. Convergence performance within the function call limit actually improved with MIMIC as the percentage of active links increased.

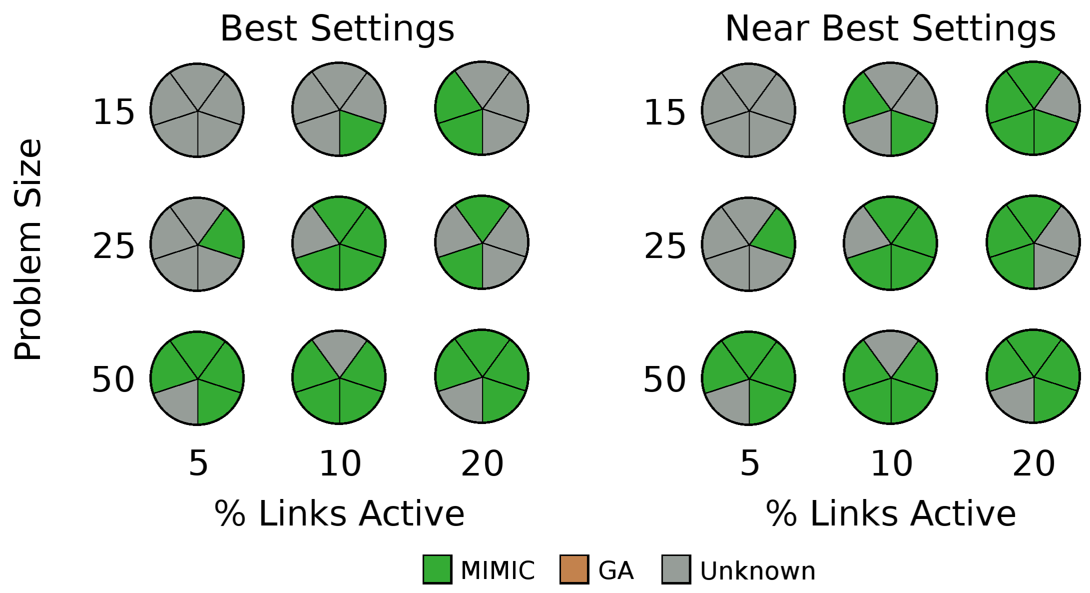


Figure 58: Percentage Not Converged Performance over Problem Set (Best and Near Best Settings)

4.1.3.4 Run-time Comparison Between Optimizers

The current modeling of pair-wise interactions that enables improved function call and convergence reliability comes at a cost of run-time performance. The Genetic Algorithm solidly performed better than MIMIC in run-time performance; even when the GA failed to converge at all and needed to hit the two million function call limit. The many cases where the GA failed to converge, for problems of large size, had MIMIC and the GA with a similar run-time performance, see Figure 60.

The 25 element problems had instances of equivalent run-time between MIMIC and GAs that was due to the fact that MIMIC performed so much better as far as function call use and convergence that it was able to converge with fewer iterations. For cases of light coupling, genetic algorithms was clearly superior in execution speed.

The averaged run-time performance in Figure 61 shows that though there are several configurations where the run-time performance overlaps with MIMIC; the GA is the faster method to reach the function call limit. The function call used for

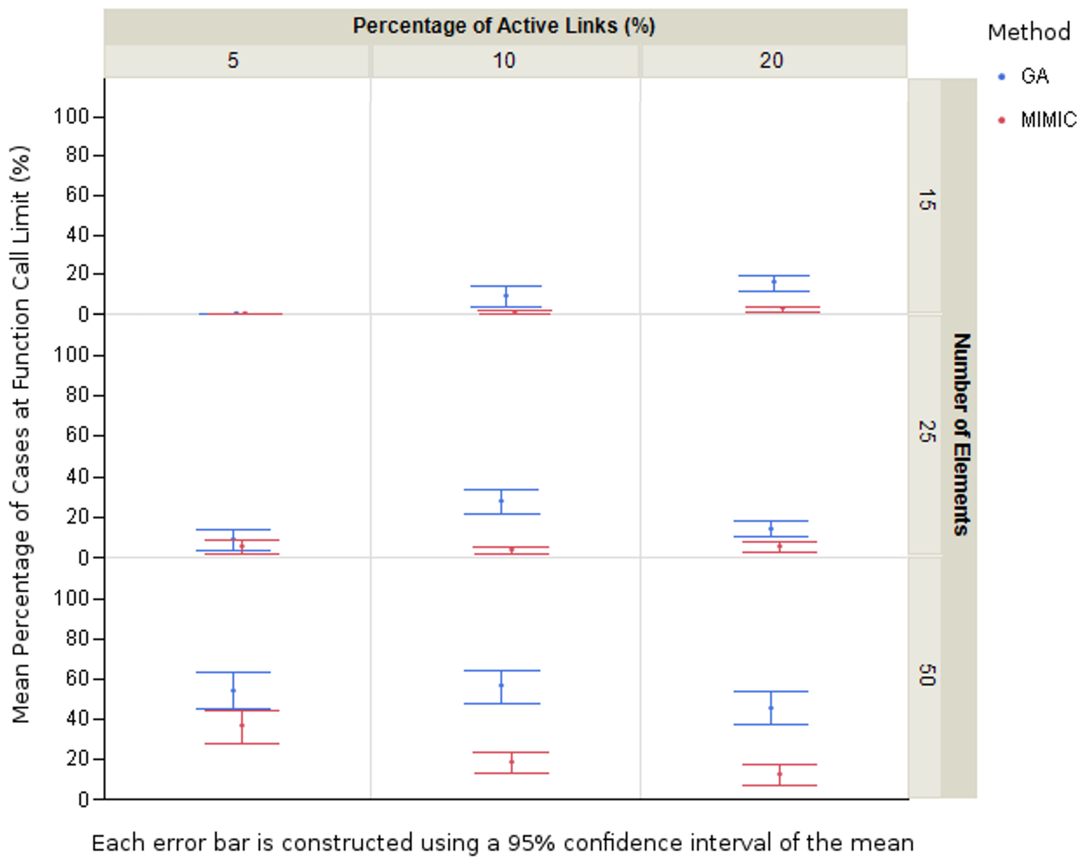


Figure 59: Mean Percentage Not Converged over Problem Set (Near Best Results included in Average)

this study was, by design, one that took only a few milliseconds to calculate.

Table 16 breaks down the time spent by the MIMIC algorithm on problem 4, which is representative of the other problems with 50 elements and 10% of links active. It is provided to display how the run-time overhead of the algorithm spends over 90% of the time on items that are not related to function call evaluation. Function call evaluations compose only around 9% of the algorithm run-time. One important point to keep in mind is that the absolute time used by the overhead (90%) will stay constant regardless of the cost of the fitness function. The calculation of the mutual information does not take any longer if a function call takes one second or one minute. Therefore, as the cost of evaluating the fitness function grows, the constant overhead will decline

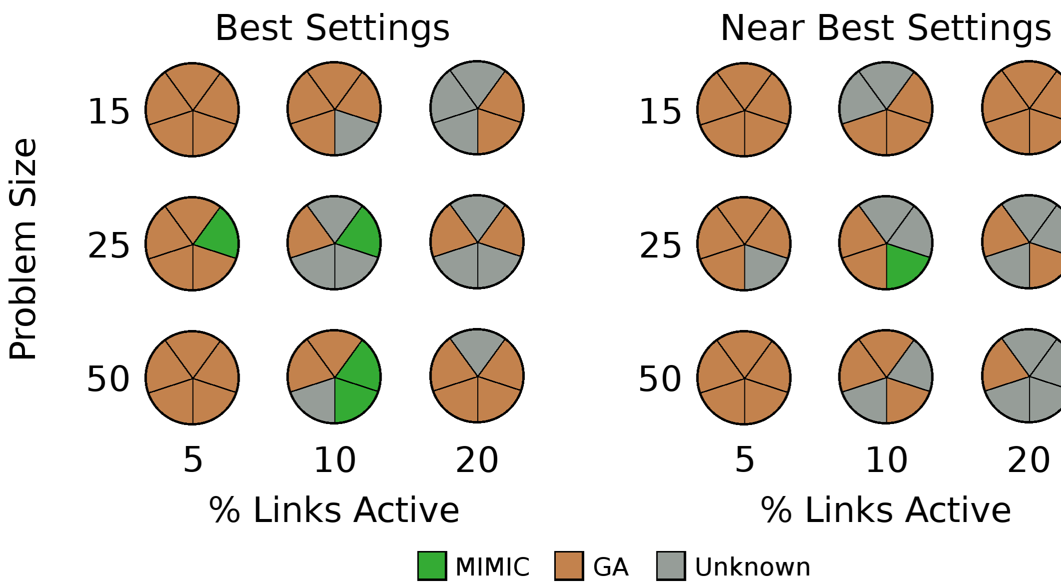


Figure 60: Run-time Performance over Problem Set (Best and Near Best Settings)

Table 16: Run-time Breakdown of MIMIC (50-Elements, 10% Links Active)

Category	**Percent Run-time**
Time to Create Initial Candidate List	<<1%
Time to Evaluate and Sort Candidates	9%
Time to Calculate Probabilities	4%
Time to Calculate Mutual Information	9%
Time to Calculate Maximum Spanning Tree	6%
Time to Generate New Samples from Tree	72%

in importance. The method that most reliably converges, using the fewest function calls, will be the preferred method. These results have shown that MIMIC would excel on medium and large static decomposition problems and relax computational constraints on function call development.

In call cases examined, MIMIC used lower mean number of function calls while Genetic Algorithms typically had a faster run-time. The function call run-time cost that would allow MIMIC to have a shorter total run-time was 11 ms on the GPU hardware specified earlier. Comparing a user's GPU hardware to the hardware used in

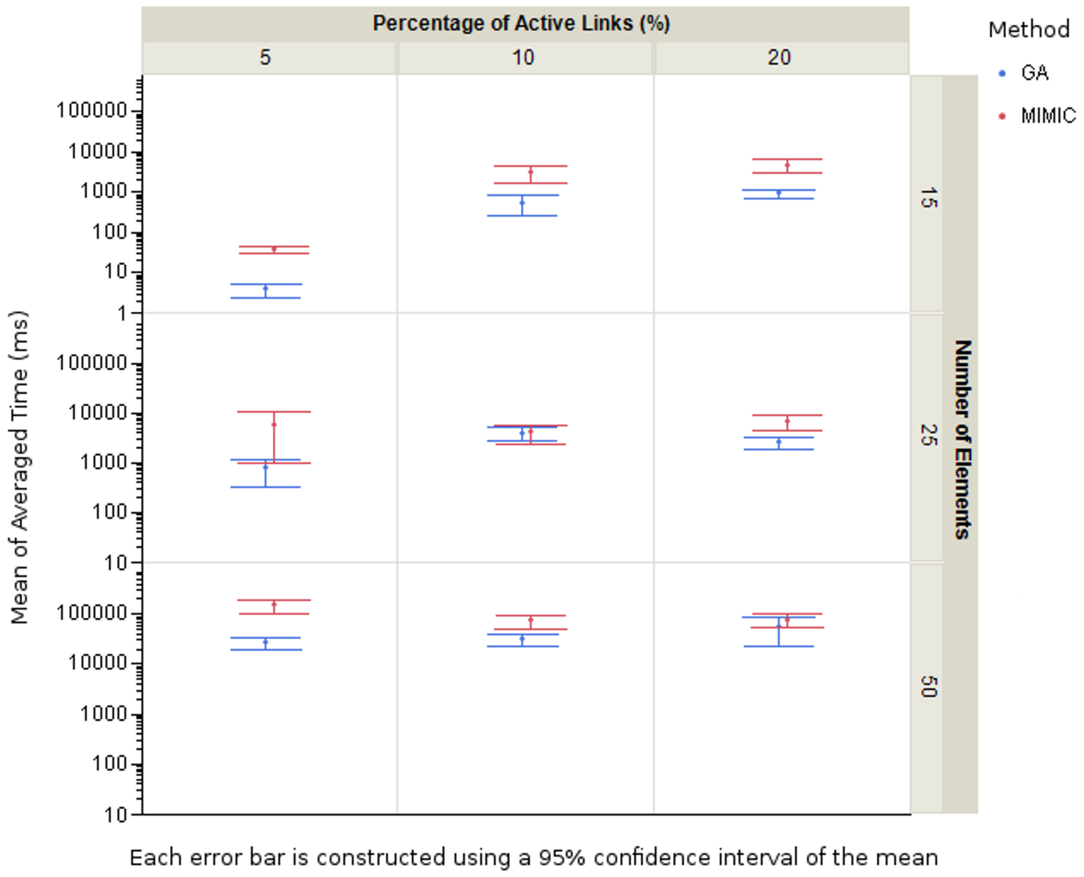


Figure 61: Mean Run-time over Problem Set (Near Best Results included in Average)

this study would allow conversion of the crossover function evaluation cost to other hardware systems. When the function call cost was above that number the lower number of function calls required by MIMIC would result in a lower total run-time. This situation will be common for any non-trivial fitness function call such as a metric that requires a past state stored in memory to calculate the utility of the current state. Any hybrid metric that uses some discipline information, such as averaged run-time of an analysis, could cost 100's of milliseconds by running an analysis and dividing that run-time by the number of function calls that can use that data.

Beyond run-time considerations, a more costly analysis is acceptable if the method is less prone to falling into local minima. This analysis has shown that MIMIC is better than Genetic Algorithms at avoiding local minima over the region of problems tested by this study. This is useful as the rearrangement of DSM problems, trajectory design, technology selection, and many other applications in aerospace are multi-modal in nature.

4.2 Generalizing Static Results with Automatic Decision Trees

A more detailed view of these results is possible through the use of additional problem descriptive metrics. A Machine Learning Open-Sourced library Weka [34] has been found that can perform the automatic creation of decision trees given a training data set. The accuracy of decision tree generation is demonstrated through cross-validation in Section 2.3.3.3. The data set required by this method was generated from the comparison cases that were generated during the performance comparison between Genetic Algorithms and MIMIC. The mutual information between the descriptive DSM metrics and the performance of the methods will be used to determine the most valuable metrics. The most pertinent metrics from Table 17 will be discovered by the decision tree generation algorithm automatically. Instead of the current practice of only showing decomposition methods applied to a specific DSM problem, method performance over a region of the DSM problem space has been examined. A decision tree for which static method to apply to the DSM problems examined has been created.

Table 17 contains a list of the metrics calculated for every problem that was randomly generated. These potential metrics are investigated to see if they might explain the behavior of MIMIC over the range of problems investigated. Items such as number of feedback or feedforward links initially in the DSM were calculated. A problem with the same size and link structure can still be randomly reordered to a different starting condition for both optimizers but that would lead to different values for the metrics in Table 17. The post-processed metrics that are most useful

Table 17: Current List of DSM Descriptive Metrics.

Metric	**Key Used in Tree**
Percentage of links that are feedback connections	perFB
Percentage feedforward connections (i.e. 1-perFB)	perFF
Ratio of feedback to feedforward connections	FB2FF
Percentage of total possible links that are active	perLinks
Ratio of number of contributing analyses to num active links	ele2Links
Number of circularly connected pairs of analyses (i.e. A→B and B→A)	numCycles
Number of feedback connections	numFB
Number of feedforward connections	numFF
Number of contributing analyses connected in the DSM	numEle
Number of active links	numLinks

for describing method performance will be discovered by the decision tree learning method. The metrics that possess a higher mutual information to the performance of either method will appear earlier in the decision tree.

The decision trees learned from the space are tested using 10-fold cross-validation to estimate the performance of the trees on untrained regions of the DSM problem space. Cross validation works by separating the set of samples into a number of groups (i.e. for 5-fold there are 5 groups). In figure 62, five separate models are created from the same data and each is validated by the data unused in training the model. The validation error is averaged between the models created to provide a final error estimate.

The results from 45 DSM problems were separated by 10-fold cross validation. Forty one DSM problems were used for the tree model while 4 were reserved for model validation. The size of the validation set is calculated by rounding down from a tenth of 45. This created a total of 10 different models from the data, each validated against data that it was not trained on, providing the averaged performance of a generated

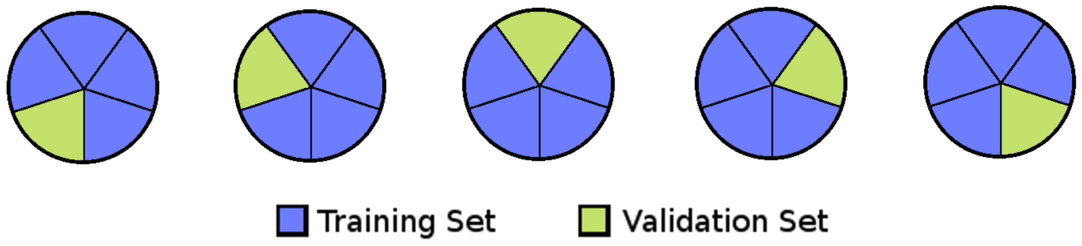


Figure 62: 5-Fold Cross Validation Separation of Training and Validation Sets.

tree to new data. This measure of averaged accuracy is provided for the decision trees provided.

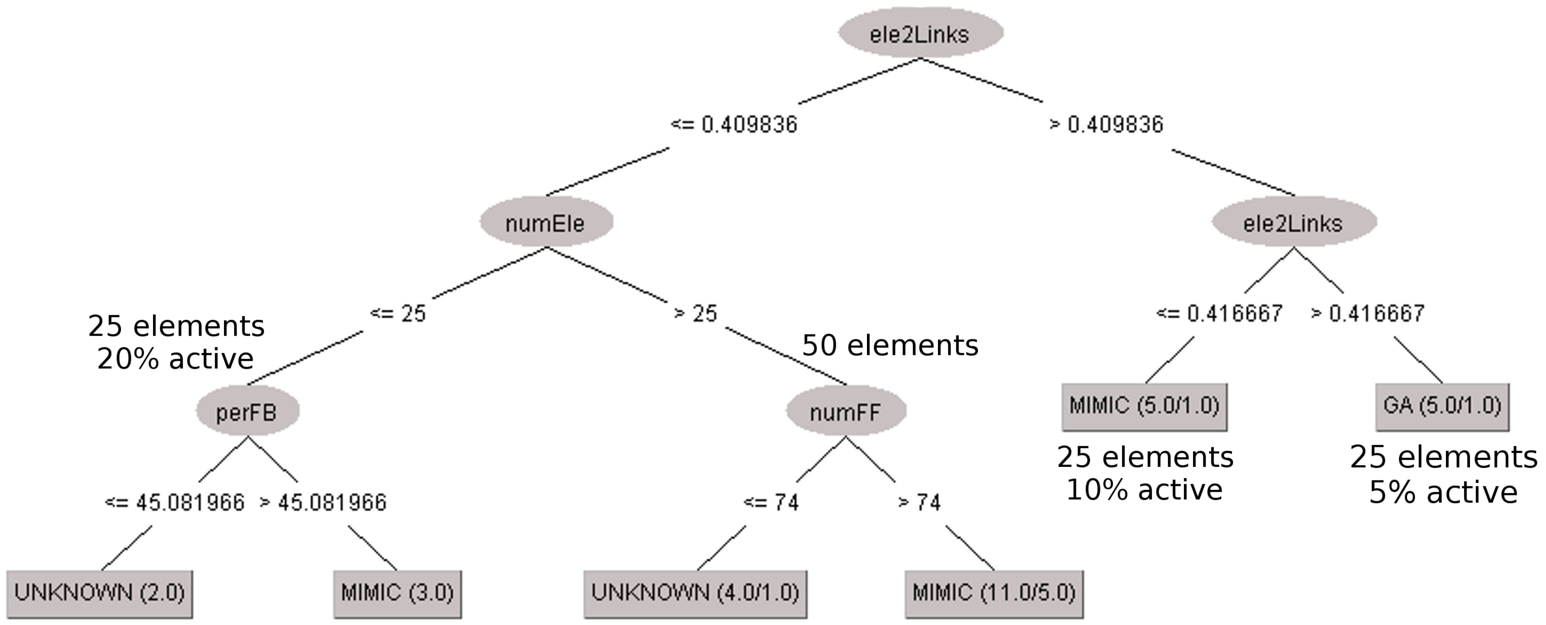


Figure 63: Decision Tree for When to Use Either Method with Best Settings (58% Accuracy using 10-Fold Cross Validation)

The decision tree generated from the results using the best settings, Figure 63, demonstrates the most valuable metrics found to classify which method should be used on a problem. The number of samples that reached each leaf in the tree and the number of samples from the training set that were misclassified are shown in the leaf labels i.e. (total samples / number misclassified). The most valuable metric when using the best settings found was the number of analyses divided by the number of active links (i.e. ele2Links). This value is unique between each of the nine configurations. The tree branches to the right in Figure 63 if the user has a 25 element problem with 10% or fewer links active. The branch to the left is taken for all other problem configurations.

The numEle node on the left branch of the Figure 63 decision tree reveals something of interest regarding the performance of the MIMIC algorithm. At this level of the tree, the two simplest problem configurations have already been classified. This branch of the tree deals with how the remaining region should be classified between either MIMIC or UNKNOWN (i.e. the null hypothesis that the methods have equivalent performance is not disproven by experimental data). The percentage of links that are feedbacks and the number of feedforward links were both metrics automatically selected as helpful to classifying which method would show better performance.

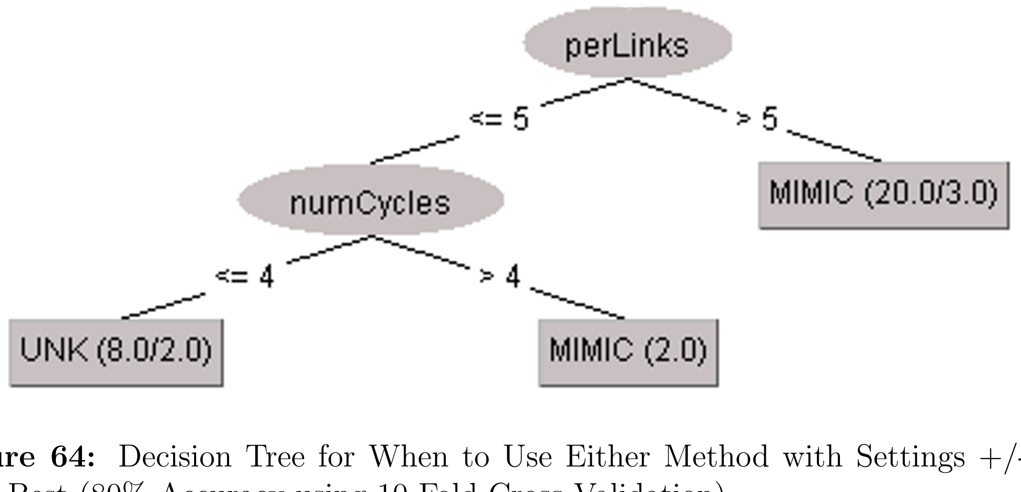


Figure 64: Decision Tree for When to Use Either Method with Settings +/-20% from Best (80% Accuracy using 10-Fold Cross Validation)

A more practical decision tree, Figure 64, is one that includes the near best setting performance. This decision tree only assumes that the user is able to fall within 20% of the best found settings for the method. Using cross validation, this tree was able to obtain an 80% predicted accuracy on data that the tree was not trained against. The metric found to best reduce uncertainty was the percentage of active links; if the percentage of active links is above 5% MIMIC is the preferred method. The last metric found suggests that the number of pair-wise cycles between analyses may also be an indicator of when MIMIC will outperform Genetic Algorithms. MIMIC is strongly represented in the decision tree and is shown to be an advancement over the standard use of GAs on static decomposition problems. This assumes that one is able to come close to a set of well performing settings for either method being used. The following serves as guidance to aid in the selection of settings for either method.

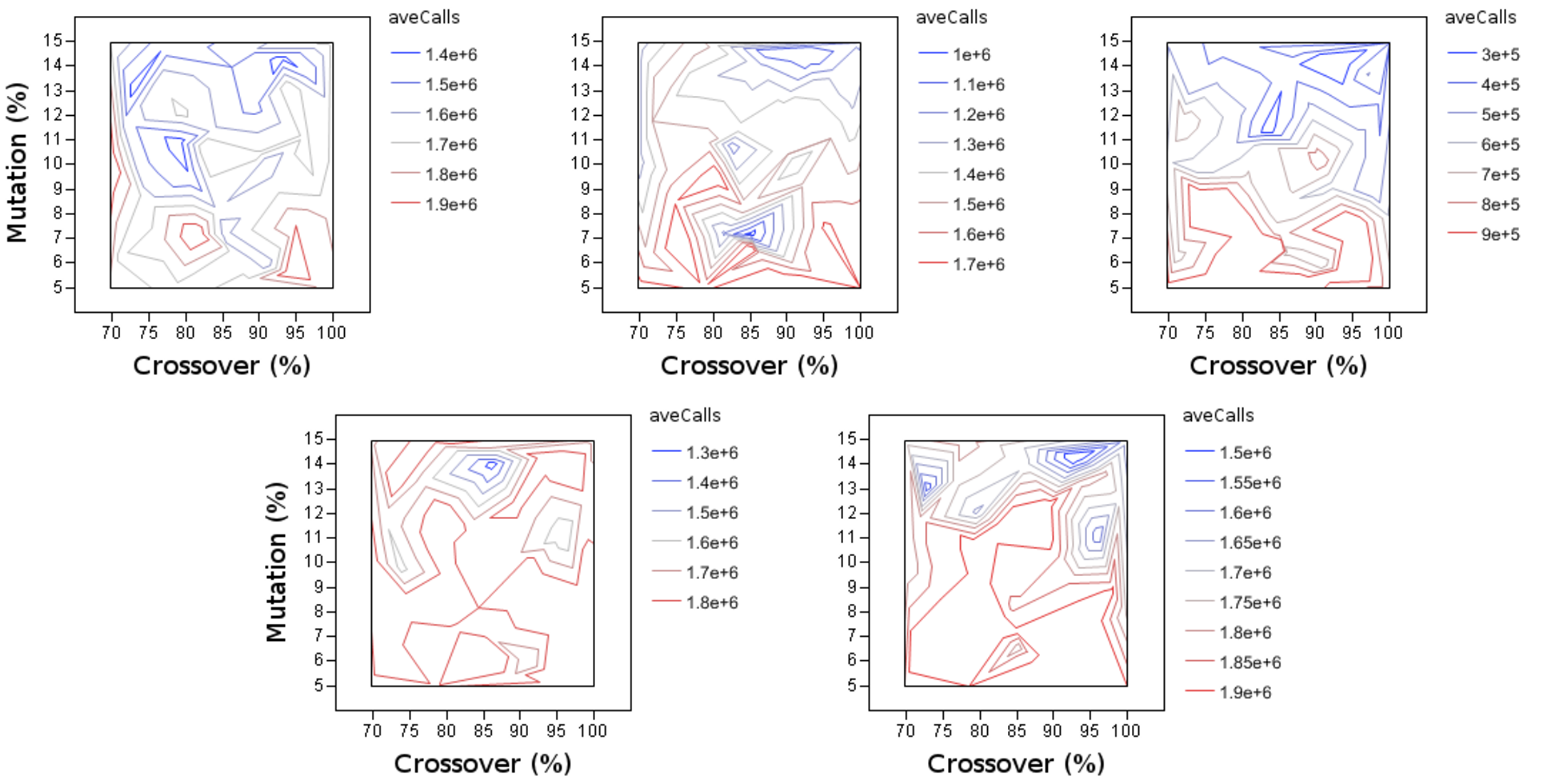


Figure 65: GA Function Call Performance over Setting Domain for Five Problems with 50 Elements and 10% Active Links

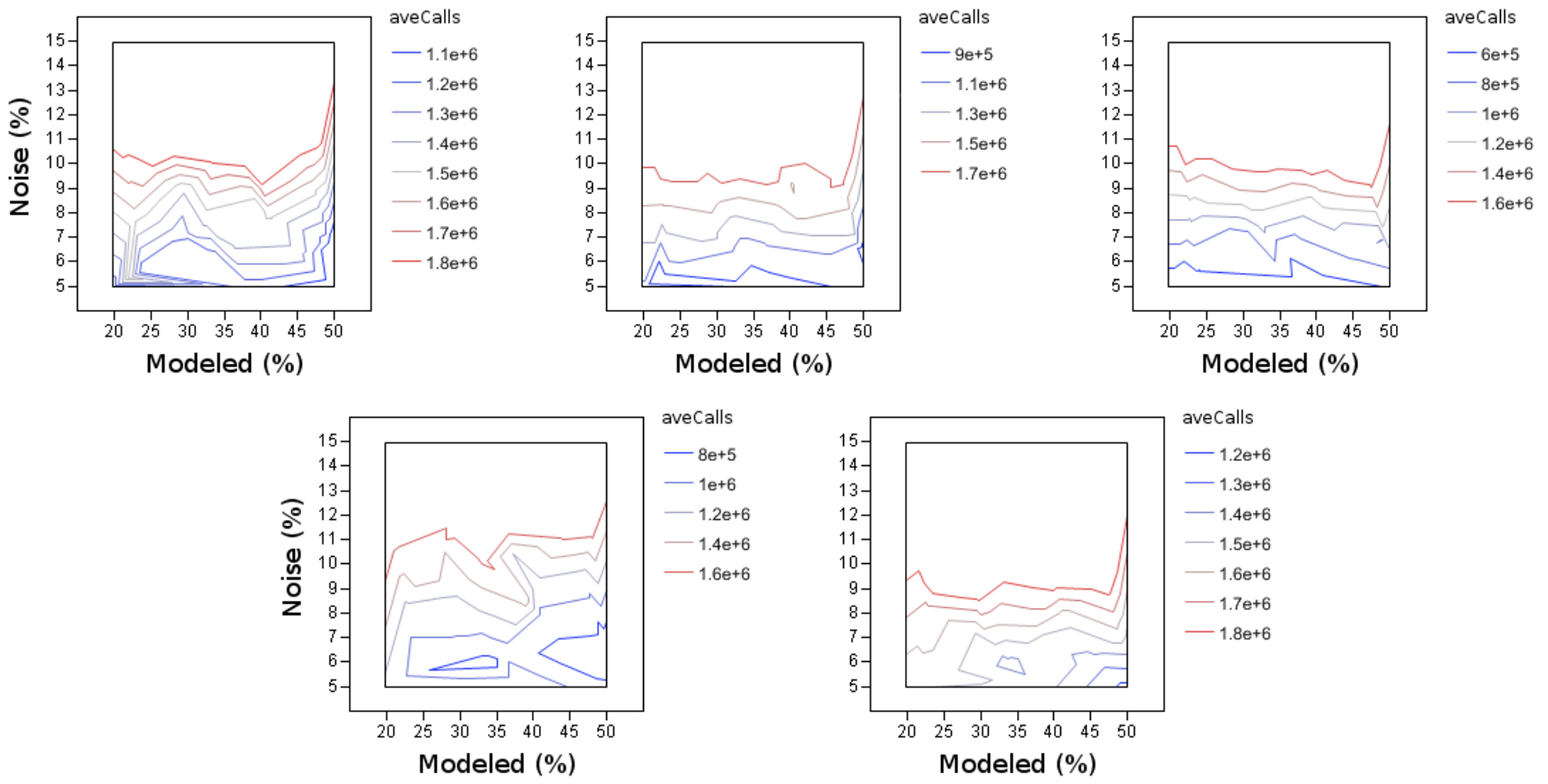


Figure 66: MIMIC Function Call Performance over Setting Domain for Five Problems with 50 Elements and 10% Active Links

The charts in Figure 65 and Figure 66 show the function call performance for each of the five problems for the problem configuration with 50 elements and 10% of active links. The performance over the range of GA settings is far more multimodal than the better behaved setting domain shown for MIMIC. This initial search of the MIMIC setting domain was necessary to realize that the noise parameter should be set in the region near or below 5% for this problem. The setting domain search was expanded to bound the performance at lower noise values.

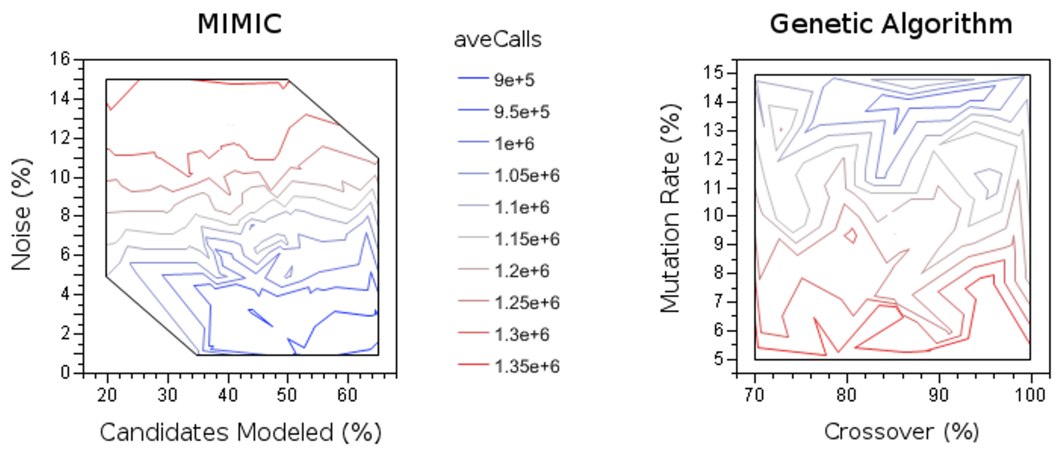


Figure 67: Method Function Call Performance over Setting Domain including All Problems

Averaging the performance of all examined settings over all problems provides Figure 67. The graph provides guidance to MIMIC users who wish to set reasonable values for either noise or the percentage of candidates that should be used for modeling. The performance of GA settings is more complicated but still implies that larger mutation rates and the neighborhood of 90% crossover is reasonable for this type of problem.

4.3 Summary

This work has served to introduce a new method for static decomposition called MIMIC that seeks to explicitly model problem structure at each iteration to reliability converge to high performing solutions using fewer function calls. A mixed Latin-Hypercube and 2-level Full Factorial design was created to search the domain of settings. Five randomly generated problems were created from six different configurations of difficulty. To run the cases within a reasonable amount of time, both methods were parallelized and run on a cluster of GPU processors. The recent advances in GPU computing were leveraged to deliver a comparison uncommon in the literature.

MIMIC was shown to strongly out perform Genetic Algorithms on the number of function calls required to reach a specified target value and was less prone to fall into local optima. The Genetic Algorithm was able to out perform MIMIC on the run-time required for loosely coupled problems (i.e. 5% active links). A run-time breakdown of the MIMIC algorithm showed that the greater run-time required by MIMIC for these problems came from a computational overhead that was not sensitive to the cost of a fitness function. The fitness function used in this comparison study took milliseconds to compute. Based on the averaged convergence behavior seen though this work, as the cost of the fitness function increased, MIMIC would perform with better run-time performance for even loosely coupled problems. Even with the inexpensive function call used in this study, the run-time performance is competitive with Genetic Algorithms over large and strongly coupled problems due to MIMIC's speed of convergence through the efficient use of function calls. This demonstrates MIMIC's strong potential for large and highly coupled problems and for problems with a large function evaluation cost.

The performance of the MIMIC algorithm was also shown to be more robust than Genetic Algorithms when using settings within 20% of the best settings found for either optimizer. This is important as the ideal settings to use on a problem are never known in practice. Getting near the ideal setting for a Genetic Algorithm was shown to provide performance that was less predictable than when using near-best settings with MIMIC. Parzen-window density estimation was also shown to perform adequately to allow the method to also treat continuous variables.

Several published utility functions have been shown for reordering a problem and are normally applied by using a global optimizer. A global optimizer is generally used as this domain contains many local optima. The efficiency of MIMIC relaxes the computational constraints that have been present for researchers. The algorithm provides a marked advancement for researchers as a replacement to the standard practice of using Genetic Algorithms for utility function research.

Machine learning algorithms were utilized to automatically generate decision trees to generalize results obtained during the static decomposition comparison between MIMIC and Genetic Algorithms. Useful descriptive metrics were developed that could predict when one optimizer would be better for a given decomposition problem. The large data set generated during the optimizer comparison was used to train the decision trees. The tree structure was pruned automatically to leave a well performing structure that was tested for accuracy through 10-point cross-validation.

With imperfect knowledge of the ideal settings, the most important predictor of which method to use was the degree of coupling present in the problem. All medium and strongly coupled problems were predicted as being better served by use of the MIMIC optimizer. These results clarify the comparison charts and provide a user with descriptive metrics that could be used on other problems to determine if the problem might perform well using MIMIC, Genetic Algorithms, or if the best optimizer could not determined by this study to within a 90% confidence.

CHAPTER V

ENTRY, DESCENT AND LANDING SYNTHESIS

Conceptual design of entry, descent, and landing (EDL) systems requires models over several disciplines as well as knowledge regarding the interactions between these disciplines. To facilitate the conceptual development of an EDL mission segment and to explore the mission design space, the Planetary Entry Systems Synthesis Tool (PESST) was created. The PESST framework estimates the performance and mass of an entry system using: user-defined geometry, aerodynamics, flight mechanics, terminal descent guidance, thermal response and mass estimation models. Trade studies can be performed by parameter sweeps to gain an understanding of the design space for conceptual studies.

Several standard atmospheres are available, and either a user-defined or global reference atmosphere model (GRAM) may be used. In its present form, PESST may be applied to entry studies for Earth, Mars and Venus missions; planets for which PESST has implemented radiative heating models. This framework is broadly applicable to the conceptual study of EDL systems and is used in this work to provide a realistic application of link importance ranking for conceptual design work.

A detailed presentation of the PESST tool and each discipline model is provided along with a comparison for the Mars Pathfinder mission. The PESST framework for system level sizing and synthesis allows for the impact of technologies such as guided terminal descent propulsion to be examined at the mission design level. Closing conceptual designs about these major discipline analyses models the effects of design changes on entry mass, peak deceleration, propellant mass, payload mass, and other mission level design constraints.

This tool has been developed as part of this work to show the determination of link importance in a conceptual design DSM with realistic discipline behavior. The analysis performed in Section 3.1 showed mutual information to be a useful importance metric for conceptual design by utilizing the PESST design tool. The individual analyses that compose PESST will be described in full detail here and together serve as a contribution to EDL design synthesis.

5.1 Introduction

The Planetary Entry Systems Synthesis Tool (PESST) is a rapid conceptual design tool for entry, descent, and landing systems. This framework has the capability to estimate the performance of an entry system using user-defined geometry, hypersonic aerodynamics, flight mechanics, thermal response, and mass estimation. Earth, Mars, and Venus atmospheric models are preloaded with the additional ability to use either a user-defined or global reference atmosphere model.[44] Trade studies can be performed by parameter sweeps to gain an understanding of the design space for conceptual studies. This framework is broadly applicable to conceptual studies of EDL systems and will be used as a realistic test case for ranking discipline inter-dependence.

Figure 68 displays the interactions that the user can have with the framework. Users that desire a graphical interface can use a Java interface that writes input files for the PESST computational core. This core executable can be run either through the graphical interface or from the command line; allowing the scripting of PESST into a larger conceptual design and optimization process. The executable core has been written with Fortran 95, utilizing object-oriented programming to promote code reuse and maintainability. The desired addition of a parametric capability also placed a high priority on the rapid completion of cases. This supported complementing the strength of Java in interface design with the computational speed and libraries available for Fortran 95.

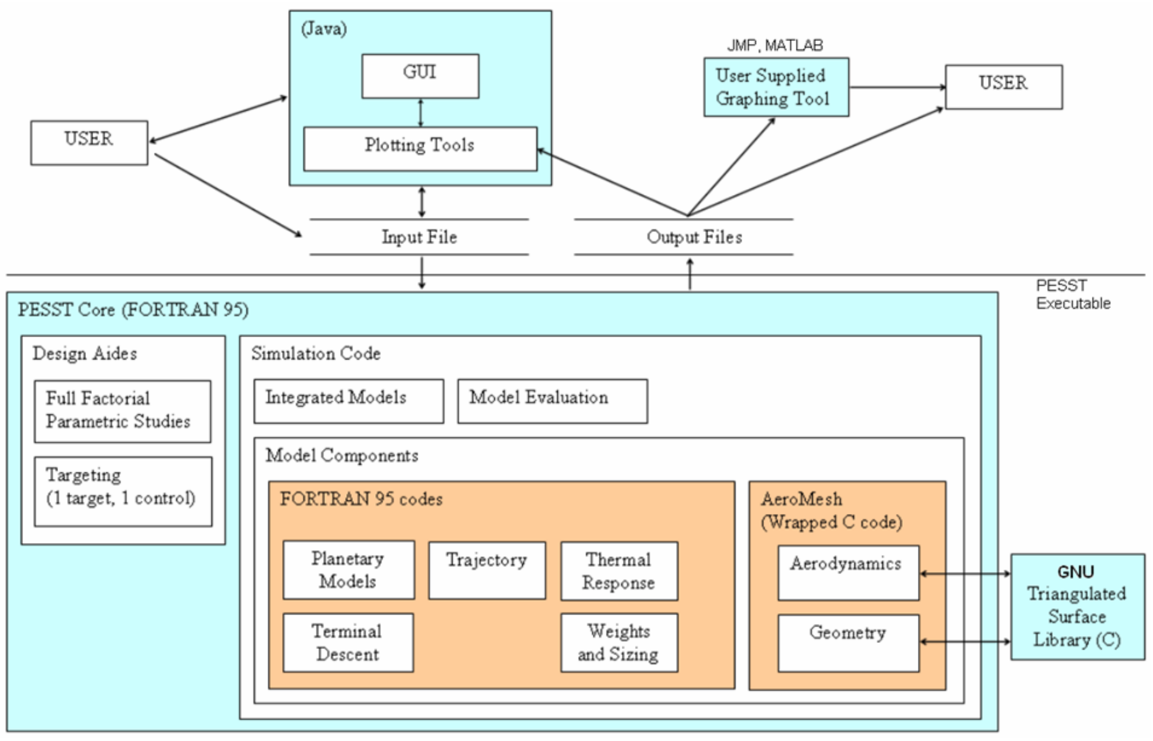


Figure 68: PESST interaction diagram.

5.2 *Discipline Modules*

PESST has seven primary modules: geometry, planetary models, aerodynamics, guidance, trajectory analysis, thermal response, and weights and sizing. Each of these modules is responsible for a separate component of the analysis and interact with one another. The modeled passing of information is shown in Figure 69 and displays the design structure matrix (DSM) for this framework.

5.2.1 System Definition

5.2.1.1 Atmospheric Models

Atmospheric models in the tool are represented through tables of temperature, pressure and density each as a function of altitude. Three atmospheric profiles are provided by the tool: a Mars atmosphere determined from Viking measurements [84], the 1976 U.S. Standard Atmosphere for Earth [65], and an averaged model for Venus

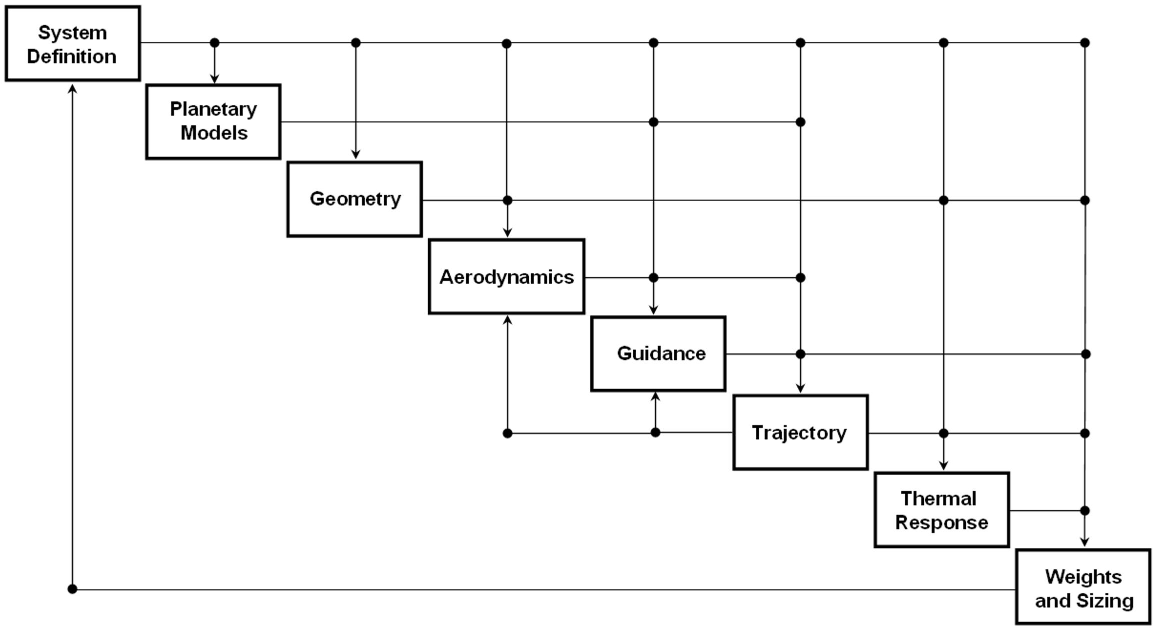


Figure 69: The Design Structure Matrix for PESST.

from Pioneer-Venus data and several of the Russian Venera missions [68]. The reader is referred to these references for the assumptions applied by these models.

PESST has the capability of running a GRAM model and will generate an atmospheric profile for a user-provided entry latitude, longitude, and date. The tool will then use this profile for the trajectory sequence through the atmosphere. The user can also manually specify and modify the atmospheric files used.

5.2.1.2 Geometry

Analytical expressions exist for the hypersonic aerodynamics of capsules, sphere-cones and biconic shapes. These expressions are computationally quick but regions of the vehicle that are shadowed from the flow can be a challenge to characterize analytically. The reader is referred to Regan and Anandakrishnan [77] for expressions on geometries with an unshadowed forebody. The work of Grant and Braun [32] is a reference for analytic expressions that characterize vehicles with a shadowed area in the forebody region.

The PESST aerodynamics module is designed to utilize a panel-based method to determine the hypersonic aerodynamics of a shape instead of using an analytical expression. The benefit to paneling a shape is the ease in determining the shadowed region of the geometry. A panel is considered shadowed when there is not an acute angle between the normal vector for the panel and the freestream velocity vector, v_∞. Vehicle shapes with windward pointing panels, that are shadowed by another body-region in a non-convex body, would not be correctly modeled by this rule, Figure 70.

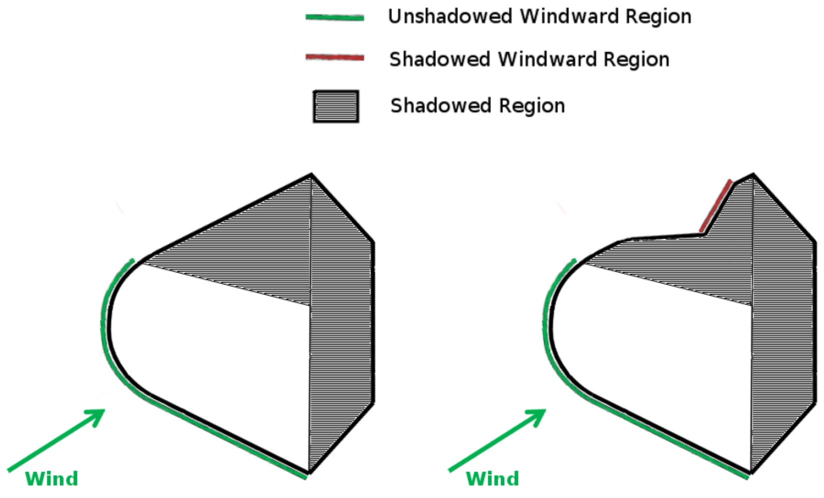


Figure 70: Example of Windward and Shadowed Regions.

An unstructured triangular meshing algorithm was utilized to panel the surface of all shapes examined by the framework. The panel-based mesh allows for one general aerodynamic treatment for both standard and complex entry bodies. Geometrical meshing and aerodynamic calculation takes less than 5 percent of the current execution time involved in a run which is normally dominated by the time to size the thermal protection system (TPS).

The mesh of panels for both the standard and arbitrary bodies are generated by the Open Sourced GNU Triangulated Surface (GTS) Library. This LGPL Library can generate a mesh from a descriptive equation or accept an input geometry file.

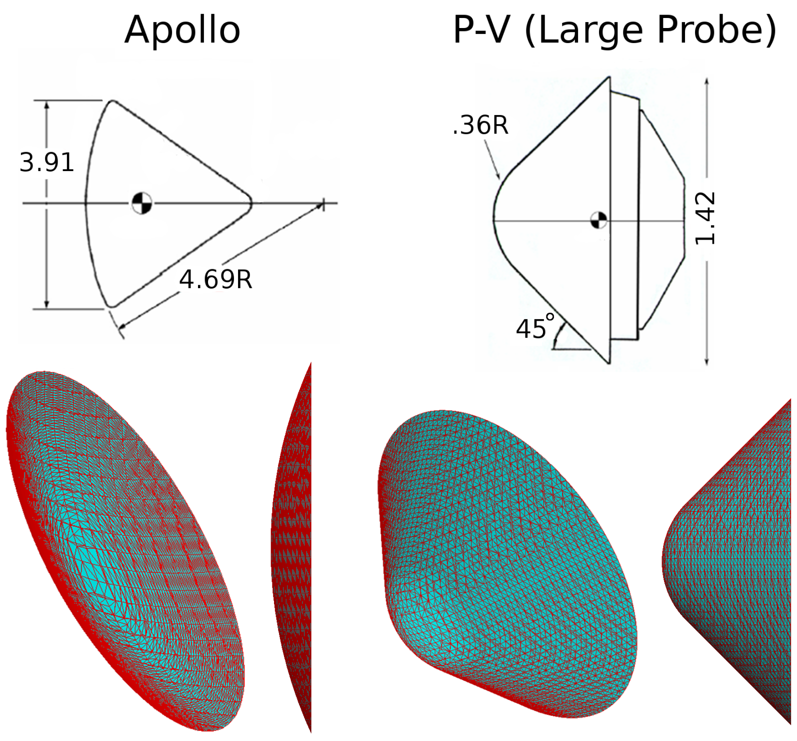


Figure 71: Meshed Geometry for Example Entry Missions.

A user can describe their entry body geometry using a CAD package and save the drawing in the common stereolithography (STL) format. The STL format contains a description of the body surface in triangular panels and is available through most CAD programs. A script from the GTS library converts the STL format to the GTS format of triangular panels used by the library. The PESST aerodynamics module utilizes the grid generated by the GTS library to calculate hypersonic aerodynamics for the described geometry.

The geometry module enables the aerodynamic method by paneling a shape into a large number of flat plates. When a user-defined mesh is not specified through the use of a STL file, the shape of the paneled surface mesh is described by providing a 3-D field equation to the GTS Library. For example, the equation for a sphere, $x^2 + y^2 + z^2 - R = C$, describes a 3-D surface mesh. The shape equations within PESST specify surfaces where the constant $C = 0$.

Equations of this nature were created for capsules, sphere-cones, and biconic shapes. The user input of forebody information provides the needed inputs for these equations to properly specify the shape of their surfaces (see Figure 71 which shows the required information to generate the displayed meshes). The GTS Library samples from the space, places points where it finds that the field equation is equal to zero, and connects the generated points to form a triangular mesh over the discovered surface of the body. This makes the addition of a new standard entry shape a simple matter of formulating the proper equation to describe the surface shape. The point discovery and mesh generation are so handled by the specialized library and are applied to each available entry body option. A STL mesh can be equivalently provided by the user for the aerodynamics module.

5.2.2 Aerodynamics

PESST enables the use of either classical or modified Newtonian aerodynamics to characterize the vehicle's aerodynamic coefficients. For Newtonian theory, all of the momentum from a fluid particle pointed normal to the surface of a plate is transferred to the plate while any momentum tangential to the plate is left unchanged. Inappropriate for the subsonic flows that Newton originally designed it for, the theory has been found to serve as an excellent approximation for hypersonic flow where these assumptions are more accurate.[2] The hypersonic behavior of a flat panel is well understood by the use of this theory. By paneling a vehicle's surface, treating each flat panel individually, and adding the accumulated forces together; the hypersonic aerodynamics for arbitrary shapes can be approximated. The assumption placed on these paneled shapes is that windward facing panels are not shadowed by another region of the entry body, Figure 70.

The aerodynamic pressure coefficient along the shadowed side of an entry vehicle is assumed to be zero. This means that only panels that are facing the flow should be evaluated. The triangles formed from the geometric meshing module are each treated as an individual panel for the Newtonian aerodynamic approximation; each windward facing panel is considered as contributing to the hypersonic aerodynamics.

5.2.2.1 Hypersonic Aerodynamics

The pressure coefficient acting on an individual panel is shown in Equation 22 where θ is the angle of the panel surface to the oncoming flow. $C_{p_{max}}$ is set to 2 for classical Newtonian flow which is typically better for slender bodies while a modification from Lees, modified Newtonian flow, refined the model to better approximate blunt body hypersonic aerodynamics.[56]

$$C_{p_{panel}} = C_{p_{max}} \sin^2 \theta_{panel} \tag{22}$$

At high Mach numbers, the modified Newtonian approximation for $C_{p_{max}}$ can be found by using the ratio of specific heats (γ) after the hypersonic shock. The ratio of specific heats cannot be assumed as constant across the shock; especially when assuming the strong shocks typically seen during an entry.[56]

$$C_{p_{max}} = 2 - \epsilon \approx \underbrace{2 - \frac{\gamma_{AS} - 1}{\gamma_{AS} + 1} = \frac{\gamma_{AS} + 3}{\gamma_{AS} + 1}}_{\text{for large Mach numbers}} \tag{23}$$

The PESST framework estimates this value initially by assuming the classical Newtonian value for maximum C_p ($C_{p_{max}} = 2$) to calculate the aerodynamics. If modified Newtonian aerodynamics is selected, a run of the trajectory module calculates the density increase behind the shock during the hypersonic portion of the entry. An average for ($\epsilon = \rho_{BS}/\rho_{AS}$) is used to update the maximum possible C_p ($C_{p_{max}}$) and the aerodynamics is refined to a tolerance.

The maximum C_p has a slight Mach dependence that is discounted. C_p becomes essentially independent of Mach at hypersonic speeds, Figure 72 shows the growing insensitivity of the axial-force coefficient to Mach through the hypersonic regime.

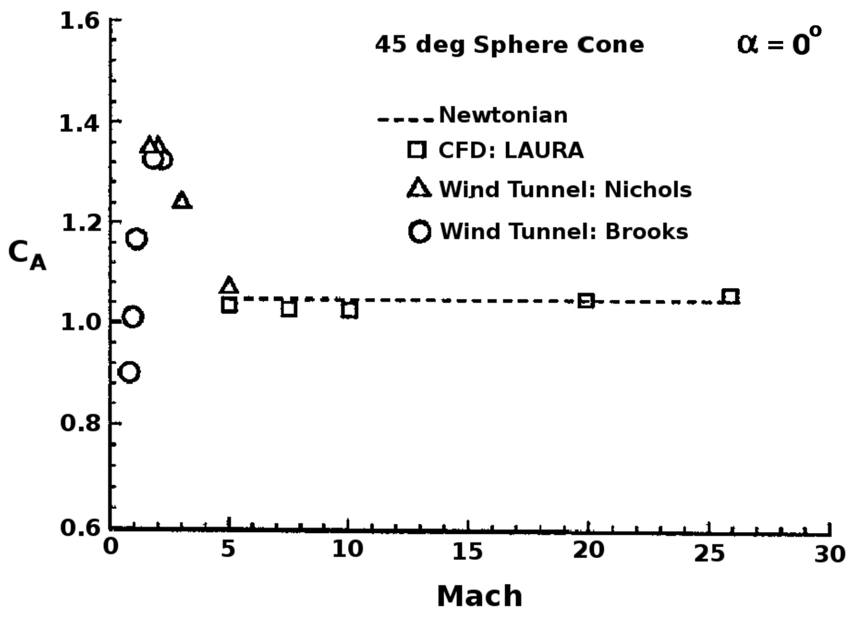


Figure 72: Axial-Force Coefficient of 45 degree sphere-cone.[67]

The axial-force coefficient is a function of pressure coefficient so the diagram also demonstrates the insensitivity of C_p at larger Mach numbers.

After evaluating the pressure coefficient for the triangular plate, the components of the force acting in the x and z directions are found to place the force into the body frame of the entry vehicle, refer to Figure 73. The normal vector for the plate ($\bar{n}$) is used to determine the component of C_p that should be applied to each component direction in the body frame. The contribution for each windward panel in the shape is then summed to obtain the contribution over the entire body, Equations 24 and 25.

$$C_{X_{tot}} = \sum_{\text{windward panels}} -C_{p_{panel}} A_{panel} \frac{n_x}{\mid \bar{n} \mid} \tag{24}$$

$$C_{Z_{tot}} = \sum_{\text{windward panels}} -C_{p_{panel}} A_{panel} \frac{n_z}{\mid \bar{n} \mid} \tag{25}$$

The forces in the body frame are divided by the reference area to find for C_A

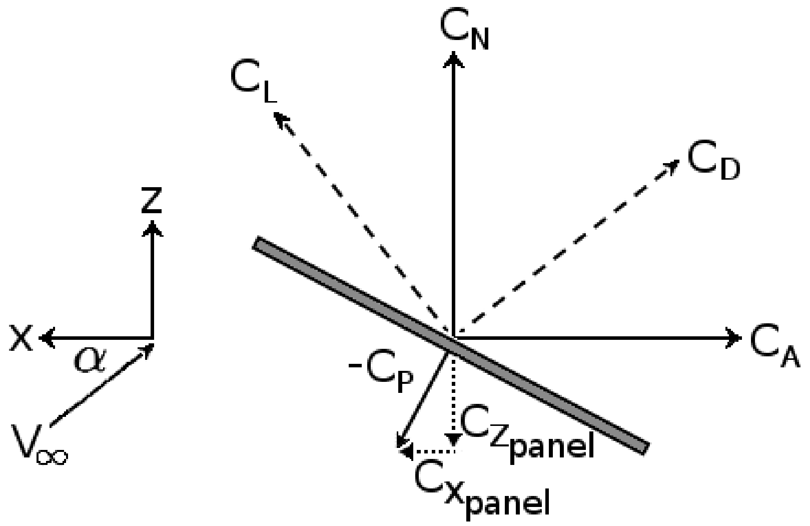


Figure 73: Reference frame for aerodynamic calculations.

which is axial to the body and C_N which is the coefficient normal to the body axis, Equations 26 and 27.

$$C_A = \frac{-C_{X_{tot}}}{A_{ref}} \tag{26}$$

$$C_N = \frac{C_{Z_{tot}}}{A_{ref}} \tag{27}$$

Forces are now translated from the body frame to the aerodynamic frame of the vehicle where the drag vector is parallel to the incoming velocity vector, taking the body to be stationary with respect to the reference frame in Figure 73. The angle of attack α is used for the rotation between the body and aerodynamic frames, Equations 28 and 29.

$$C_L = C_N \cos\alpha - C_A \sin\alpha \tag{28}$$

$$C_D = C_N \sin\alpha + C_A \cos\alpha \tag{29}$$

Users are also able to specify their own aerodynamics for this module. For example, the deployment of inflatable aerodynamic decelerators or parachutes are specified by aerodynamic files that can be activated during the trajectory analysis. The calculated aerodynamics for the vehicle are temporarily replaced by the aerodynamics from the specified files. Using the same mechanism, the users can define their own aerodynamics to override the computed aerodynamics during a trajectory evaluation.

5.2.3 Flight Mechanics

Several books and references have used planet-relative kinematic equations for hypersonic entry bodies that describe the time derivatives of longitude, latitude, radial distance, azimuth, flight path angle, and velocity.[47, 77] When using this planet-relative approach to determine the derivatives for azimuth and the flight path angle, the equations are divided by the current velocity (i.e., $\dot{AZ} \sim \frac{1}{V}$ and $\dot{\phi} \sim \frac{1}{V}$). During terminal descent, when the velocity becomes small, the trajectory equations become poorly behaved and are not valid for determining the thrust vector to command. PESST has moved away from this approach after incorporating terminal descent guidance algorithms into the framework.

In the present implementation, the equations of motion are switched to a planet-centered inertial reference frame, this will be referred to as the inertial frame. When converting to/from this reference frame the transformation matrix is considered as being solely a function of the planet's rotation about a stationary z-axis during the entry. The vector formulation for the equations of motion in the inertial frame is well behaved, over all velocities, and it does not experience the equation singularities that can appear with certain angles in planet-relative equations.

$$\begin{bmatrix} \dot{\bar{r}}^I \\ \dot{\bar{v}}^I \\ \dot{m} \end{bmatrix} = \begin{bmatrix} \bar{v}^I \\ \sum \bar{F}^I/m \\ -\left|\bar{T}\right|/(g_o\, I_{SP}) \end{bmatrix} \tag{30}$$

PESST now utilizes three degree-of-freedom (DOF), with bank angle modulation, equations of motion to determine the time history of the entry system's state, Equation 30. The trajectory is propagated using a variable-step 4th-order Runge-Kutta integrator with 5th-order error truncation over a fixed time-span from a set of planet relative initial conditions (altitude, velocity, flight path angle, azimuth angle, latitude, longitude, angle of attack, and bank angle) until the end of the simulation. The starting condition is transformed to inertial space and the movement of the vehicle is integrated inertially. As the PESST framework converts angles to/from inertial coordinates, the framework does not experience angular singularities; it is effectively post processing to get the flight path angle or any other angle.

This treatment allows for forces to be easily added or removed from the entry vehicle; as only the inertial summation of forces on the body is required for each time step through the integration. The forces tracked by the framework, shown in Figure 74, are: drag, lift, the weight vector due to gravity, and a thrusting vector if guidance is active. The method is more maintainable and extendable than the use of angular derivatives. For example, the addition of wind effects could be enabled through the manipulation of the velocity vector. Alternatively, a buoyancy force, useful for Titan entries, could also be easily appended to the summation of forces.

$$L = -q\, C_L\, A_{ref} \tag{31}$$

The lift magnitude is determined from Equation 31. The reference area of the vehicle is A_{ref} and q is the dynamic pressure. The direction of the lift vector is perpendicular to the velocity vector and is rotated, about the velocity vector, by the bank angle specified for the vehicle.

$$\bar{D} = -q\, C_D\, A_{ref} \frac{\bar{v}^{rel}}{|\bar{v}^{rel}|} \tag{32}$$

The vector for vehicle drag is parallel to the planet relative velocity vector and is

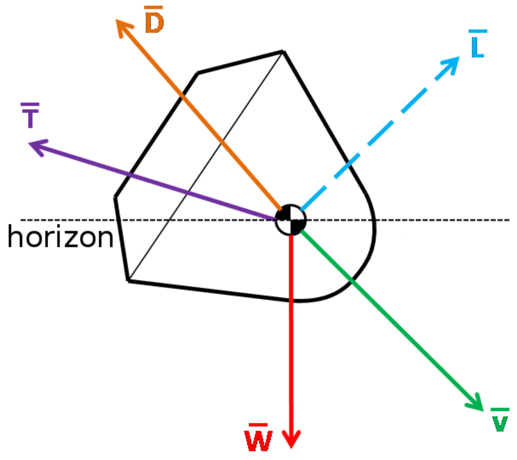


Figure 74: Entry system free-body diagram.

specified by Equation 32.

$$\dot{m} = \frac{-\left|\bar{T}\right|}{g_o \, I_{SP}} \tag{33}$$

The momentum thrust equation is used to estimate the instantaneous mass loss during propulsive maneuvers, Equation 33. Discrete mass drop events, such as the drop of a heat shield, are modeled by a discontinuous drop to the system mass at the drop point.

5.2.4 Terminal Descent Guidance Algorithms

5.2.4.1 Gravity Turn

PESST allows for the specification of a constant thrust gravity turn event. In a gravity turn, the thrust vector is in the opposing direction from the current velocity vector, Figure 75. This causes the vehicle to turn toward a vertical descent from the effect of gravitational force on the vehicle. This method targets for a specified velocity at a given altitude. A constant thrust gravity turn was utilized for the terminal descent of Lunar Surveyor and other missions.

An analytic equation for the thrust magnitude can be obtained if a small nadir angle is assumed. PESST avoids the small nadir angle assumption by numerically

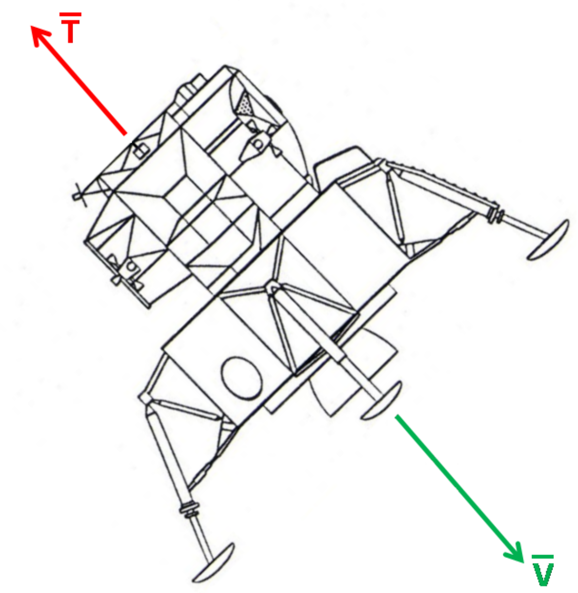


Figure 75: Gravity turn thrust orientation. Lander image from [74].

determining how well the constant thrust value meets specified targets. This is implemented by specifying an initial fixed thrust magnitude and propagating the equations of motion to determine how well that thrust value meets the velocity and altitude targets. Newton iteration is used to converge towards the fixed thrust magnitude needed for the desired final altitude and velocity magnitude. Specifying the maximum thrust magnitude for the engine would allow the calculation of the latest possible ignition time that could successfully match the velocity and altitude target.

5.2.4.2 Analytical Control Law

The work of D'Souza [24] developed an analytic control law that assumes a planar non-rotating planet with no atmosphere. This was found to be suitable for Martian applications that required a first-order model for pinpoint landing. This analytic method is an unconstrained energy-optimal propulsive terminal descent algorithm that meets the necessary and sufficient conditions for an optimal control law.[90]

$$J = \Gamma\, t_f + \frac{1}{2}\int_{t_o}^{t_f} \bar{a}^T \bar{a}\; dt \tag{34}$$

The D'Souza algorithm seeks to minimize the performance index shown in Equation 34. For the current PESST implementation, the weighting on the final time Γ has been set to zero. It has been shown that the control law which minimizes this performance index is given by Equation 35, given the vector definitions from the vector Equations in 36.[24]

$$\bar{a} = -4\frac{\Delta\bar{v}}{t_{go}} - 6\frac{\Delta\bar{r}}{t_{go}^2} - \bar{g} \tag{35}$$

$$\Delta\bar{r} = \begin{bmatrix} r_x - r_{x_f} \\ r_y - r_{y_f} \\ r_z - r_{z_f} \end{bmatrix} \quad \Delta\bar{v} = \begin{bmatrix} v_x - v_{x_f} \\ v_y - v_{y_f} \\ v_z - v_{z_f} \end{bmatrix} \quad \bar{g} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{36}$$

The time-to-go, t_{go}, is provided as the real positive root to equation 37. This equation is solved by Newton iteration which provides rapid convergence while mitigating the numerical issues that can be associated with analytical root equations.[90]

$$t_{go}^4 - 2\frac{\Delta\bar{v}^T\Delta\bar{v}}{\Gamma + \frac{g^2}{2}}t_{go}^2 - 12\frac{\Delta\bar{v}^T\Delta\bar{r}}{\Gamma + \frac{g^2}{2}}t_{go} - 18\frac{\Delta\bar{r}^T\Delta\bar{r}}{\Gamma + \frac{g^2}{2}} = 0 \tag{37}$$

The thrust that is commanded by the guidance algorithm at each point in time is given by Equation 38. When the guidance event is active, the thrust is computed in a closed loop sense at each timestep through the equations of motion. The reader is referred to the work of Steinfeldt, et al. for performance and accuracy comparisons with this analytic control law.[90]

$$\bar{T} = m\left(-4\frac{\Delta\bar{v}}{t_{go}} - 6\frac{\Delta\bar{r}}{t_{go}^2} - \bar{g}\right) \tag{38}$$

5.2.5 Thermal Response

Modeling of a thermal protection system (TPS) requires the modeling of an energy balance that involves some subset of the processes that occur during the ablation of the TPS material. PESST currently models three ablative materials: SLA-561v, PICA and Carbon Phenolic (FM5055). Figure 76 shows some of the energy and mass-related processes that are involved during an ablative entry. In the PESST thermal modeling, stagnation point convective and radiative fluxes will be explicitly modeled, while surface recession will be indirectly modeled by the use of an experimental correlation called the heat of ablation.

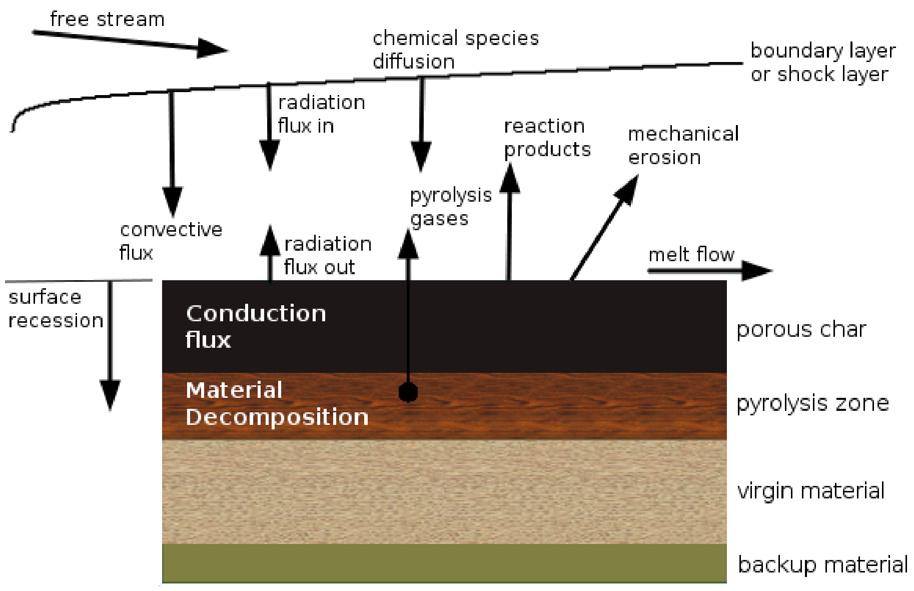


Figure 76: Processes during the ablation of a material [54].

The problem of estimating the TPS thickness required for an entry is decomposed into two sub-problems; the thickness that will be ablated during the descent and the thickness required to insulate the vehicle. Both thicknesses are assumed to consist of the same user-selected ablative TPS material. The two thicknesses are added together to estimate the composite thickness required by the TPS material to both survive ablation and retain adequate insulation for the vehicle.

5.2.5.1 Heating Environment

The Sutton-Graves heating relationships are used to estimate the degree of convective stagnation point heating.[92] The form of this relationship is shown in Equation 39 where $\bar{v}^{rel}$ is the planet relative velocity and R_n is the effective nose radius of the vehicle. The Sutton-Graves convective constant k_{sg} is a function of the composition of the atmosphere. The convective relation was designed for an arbitrary atmosphere.[92] PESST currently uses pre-computed values for Earth, Mars and Venus but future work could dynamically calculate the convective constant from the user's specification of the atmospheric composition.

$$\dot{q}_{conv} = k_{sg} \frac{\sqrt{\rho} \left|\bar{v}^{rel}\right|^3}{\sqrt{R_n}} \tag{39}$$

$$\dot{q}_{rad} = C \, R_n^{a_{ts}} \, \rho^{b_{ts}} \, f(\left|\bar{v}^{rel}\right|) \tag{40}$$

A general radiative relation for an arbitrary atmosphere is an open problem. Radiative relations for Earth and Mars are supplied by the work of Tauber and Sutton, the form of the relation is shown in Equation 40.[94] The constants C, a_{ts}, b_{ts}, and a function of the planet relative velocity $\bar{v}^{rel}$ are defined for Earth or Mars. A pre-publication radiative expression for Venus was provided by Dr. Tauber through correspondence. The data he used to generate the Venus relation can be found from the work of Page and Woodward.[72]

The total stagnation point heating is considered as the summation of the radiative and convective stagnation point heating, Equation 41.

$$\dot{q}_{tot} = \dot{q}_{conv} + \dot{q}_{rad} \tag{41}$$

5.2.5.2 Thickness Ablated

The heating from the surface into the material is required to size the insulation thickness required from the TPS material. Equation 42 is an energy balance where the left hand side of the equation represents energy (heat) traveling into the material and the right hand side represents the other processes occurring at the ablating surface.

$$-k\frac{\partial T}{\partial x} = -\dot{q}_{cw}\tfrac{H_r - H_{air}^{T_w}}{H_r} + \tag{42}$$

$$\sigma_{SB}\,\varepsilon\,T_w^4 + \tag{43}$$

$$\rho\,\dot{s}\,\Delta H_v + \tag{44}$$

$$\rho\,\dot{s}\,\eta\,\left(H_r - H_{air}^{T_w}\right) \tag{45}$$

The right side of Equation 42 shows the convective net heat flux into the surface without ablation.[22] The term from Equation 43 shows the heat flux re-radiated from the surface of the heat shield. The term from Equation 44 calculates the amount of energy absorbed through the vaporization of the ablating material. The last term, Equation 45, shows the amount of heat absorbed through the transpiration of ablation gases into the flow.[22]

$$\left(k\frac{\partial T}{\partial x}\right)_{ss} = \rho\dot{s}\,C_p\left(T_w - T_o\right) \tag{46}$$

Assuming steady state ablation, the equation for the heat traveling into the material simplifies to Equation 46. Substituting Equation 46 into Equation 42 forms Equation 47.

$$\dot{q}_{cw}\left(\frac{H_r - H_{air}^{T_w}}{H_r}\right) - \sigma_{SB}\varepsilon T_w^4 = \rho\dot{s}\,\left(C_p\Delta T + \Delta H_v + \eta\Delta H\right) \tag{47}$$

Manipulating the terms for the equation forms the definition for the experimental relation Q^*, the heat of ablation. The central terms of Equation 48 are composed of values that can be tested in a laboratory setting. It is through experimental testing that a curve for Q^* is found.

$$Q^* = \frac{\dot{q}_{cw}\left(\frac{H_r - H_{air}^{T_w}}{H_r}\right) - \sigma\varepsilon T_w^4}{\rho\dot{s}} = (C_p\Delta T + \Delta H_v + \eta\Delta H) \tag{48}$$

As a simple ablation relationship is desired, an approximation is made to the experimental curve found for the heat of ablation. The rate of thickness lost due to steady state ablation $\dot{s}$ is less than the value shown in Equation 49. $\dot{q}_{cw}$ is the rate of cold wall heating at the surface; this is taken as equal to the total rate of heating computed in Equation 41. Given a curve for Q^* vs. $\dot{q}_{cw}$ a conservative estimate can be made for the rate of thickness lost due to steady state ablation by assuming an equality in Equation 49. This calculation could be integrated throughout the trajectory to calculate the estimated total thickness lost due to ablation.

$$\dot{s} < \frac{\dot{q}_{cw}}{\rho\, Q^*} \tag{49}$$

Several experimental curves for Q^* vs. $\dot{q}_{cw}$ are shown in Figure 77.

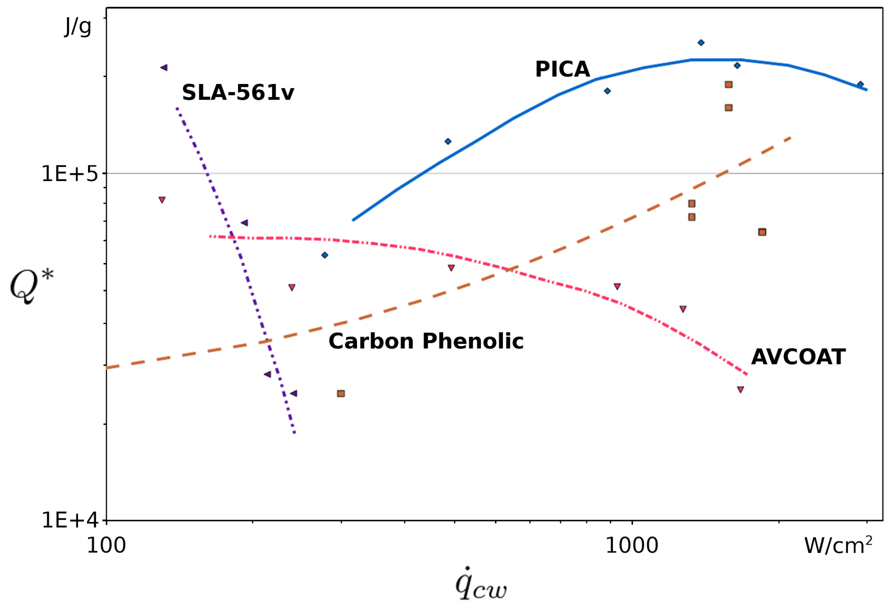


Figure 77: Heat of ablation vs. cold wall heat rate. [76, 101]

5.2.5.3 Thickness used for Insulation

The computed temperature at the surface of the heat shield serves as a boundary condition to calculate the transient in-depth temperature behavior through the material. This is used to calculate the thickness of the material required to insulate the structure behind the TPS system to a specified boundary temperature limit. The 1-D heat conduction equation where density, specific heat (C_p) and the thermal conductivity (k) vary with position is given by Equation 50.

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = \rho C_p \frac{\partial T}{\partial t} \tag{50}$$

This in depth temperature equation can be solved by either an explicit or implicit method. Here an implicit method has been used to track the response of the material to the application of surface heating. The Crank-Nicolson implicit method used is unconditionally stable and the finite difference approximation has a truncation error of order $O[(\Delta t)^2, (\Delta x)^2]$.

$$\frac{\partial \rho}{\partial t} = f(\rho, T) \tag{51}$$

The decomposition rate from Equation 51 is modeled by a three component Arrhenius relation shown by Equation 52. The constants required to use the equation for PICA and Carbon Phenolic are provided in Appendix A.

$$\left.\frac{\partial \rho}{\partial t}\right)_x = -B\exp\left(\frac{-E}{R_{ug}T}\right)\rho_o\left(\frac{\rho-\rho_r}{\rho_o}\right) \tag{52}$$

To compute the density for the composite material, the densities for each component need to be included, Equation 53.

$$\rho = \Gamma_{resin}\left(\rho_{Aresin} + \rho_{Bresin}\right) + \left(1-\Gamma_{resin}\right)\rho_{Cfiber} \tag{53}$$

Refer to the tables in Appendix A for thermal material information used by the PESST framework to perform these calculations.

5.2.6 Mass Estimation

5.2.6.1 Thermal Protection System Sizing

The ablation rate $\dot{s}$ is integrated over the trajectory to estimate the thickness lost from ablation. The stagnation heat conducted into the material is used to size the minimum thickness required to keep the interior wall of the material below a user specified bondline temperature. Both processes are considered as decoupled so the thicknesses are added to give a final uniform heatshield thickness.

$$m_{struct} = 0.08\, m_{entry} \tag{54}$$

$$m_{heatshield} = m_{struct} + A_{tps}\, thickness_{tps}\, \rho_{tps} \tag{55}$$

The heatshield mass is computed from the provided area covered by the heatshield A, the density of the heatshield ablative material ρ_{tps} (Table 18), and the computed total thickness due to the heating history of the stagnation point region.

Table 18: TPS Material Density.

TPS Material	**Density** (kg/m^3)
PICA	227.4
SLA-561v	264.3
Carbon Phenolic (FM5055)	1435.4

The structural mass of the heatshield is estimated by the relationship shown in Equation 54. The structural mass is added to form the final estimate for a constant thickness heatshield, see Equation 55.

5.2.6.2 Propulsion System Sizing

Any thrust events performed during the trajectory add to the budgeted fuel requirement by integrating the $\dot{m}$ computed through the momentum thrust equation, see Equation 33. The density for the fuel and oxidizer and O-F ratio are used to determine the volume requirements for each liquid.

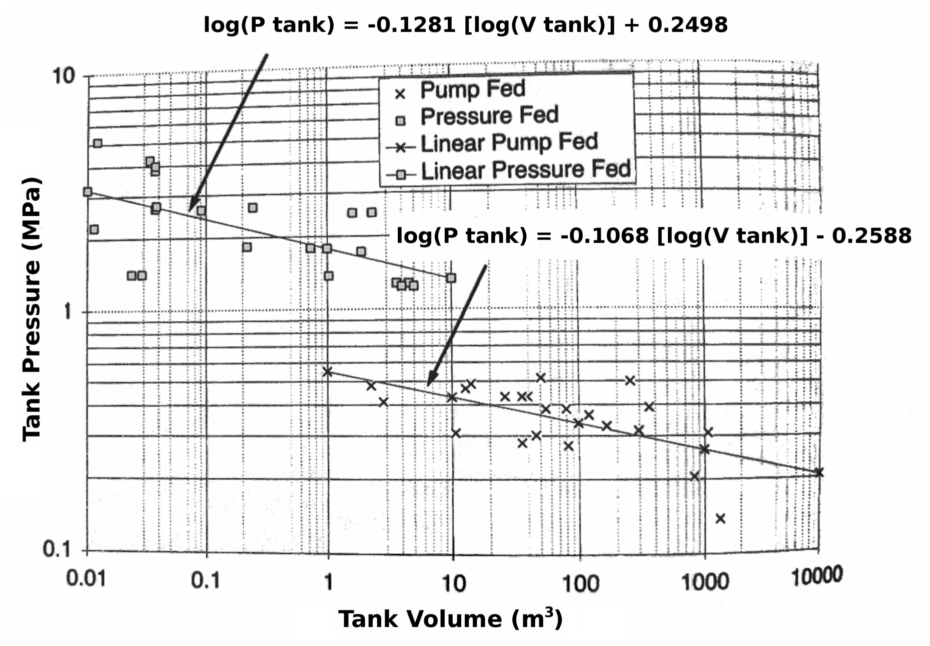


Figure 78: Tank Pressure Fits by Engine Type.[40]

The operating pressure for the tank is determined from the engine type and tank volume using the data regression in Figure 78 to historical data of different engine types. This operating pressure is used to size the volume of Helium required to act as a pressurant for this tank, in pressure-fed systems. The reader is referred to [40] for further details.

$$m_{tank} = \frac{f_{safety}\, P_{tank}\, V_{tot}}{g_0\, \phi_{tank}} \tag{56}$$

The pV/W method is utilized to calculate the tank mass without specified tank dimensions. This method requires the specification of a tank mass factor ϕ_{tank} that

Table 19: Tank Mass Factors.

Tank Material	**Tank Mass Factor** (m)
Al	2500
Ti	5000
Ti with Carbon Fiber	10000

describes the strength of the tank material. A completely metallic tank has a tank factor equal to 2500 m; with reinforcing materials the tank factor can rise to 10000 m.[40] The values used by PESST for the three available tank materials are shown in Table 19.

The pV/W method uses the burst pressure for the tank $P_{burst} = f_{safety} P_{tank}$ (Pa), tank volume V_{tot} (m^3), and tank factor ϕ_{tank} (m) to calculate for the tank mass in kilograms. The burst pressure for the tank is estimated in PESST as the operating pressure times a safety factor equal to 1.5. This method is utilized to determine the mass for the fuel, oxidizer, and potential pressurant tanks.

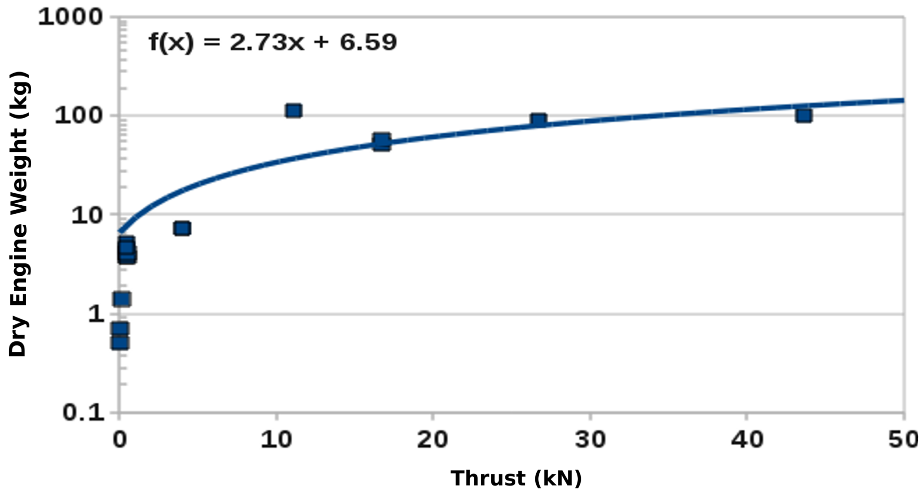


Figure 79: Sizing Fit for Monomethyl Hydrazine (MMH) Engines. Historical data from [40].

Regressions through historical engine mass data are used to size for the engine based on the engine maximum thrust and fuel type. The equations in Figures 79,80,81

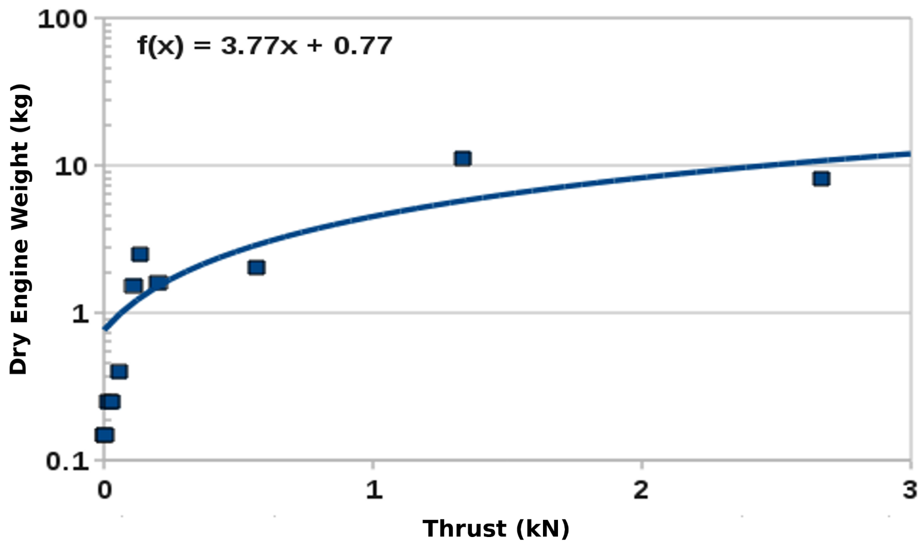


Figure 80: Sizing Fit for Monopropellant Hydrazine Engines. Historical data from [40].

use x in kN to compute the dry weight of the engine in kg on these semi-log charts.

The LOX/CH_4 engine weight data provided from [48] cited the engine wet weight, i.e. the engine weight with fuel filled lines. Six percent of the weight cited in the paper was removed to regress the estimated dry weight, Figure 81. The lines and valves for the engine are assumed to weigh as much as the engine itself.

The summation of the masses for the fuel tanks, pressurization system (pressurant + pressurant tank), engine, lines and valves is given for the propulsion system mass.

5.2.6.3 Parachute System Sizing

$$\begin{aligned} F_{max} &= C_k \, q_{deploy} \, C_D \, A \\ F_{design} &= f_{safety} \, F_{max} \end{aligned} \tag{57}$$

The parachute sizing module models an infinite mass inflation with the peak load occurring at full parachute inflation. The parachute is sized based on this peak load with a safety factor equal to 1.5. The C_k method was utilized with a $C_k = 0.6$ for DGB parachutes and the area $A = \pi \left(\frac{D_o}{2}\right)^2$. The material properties for kevlar are used to

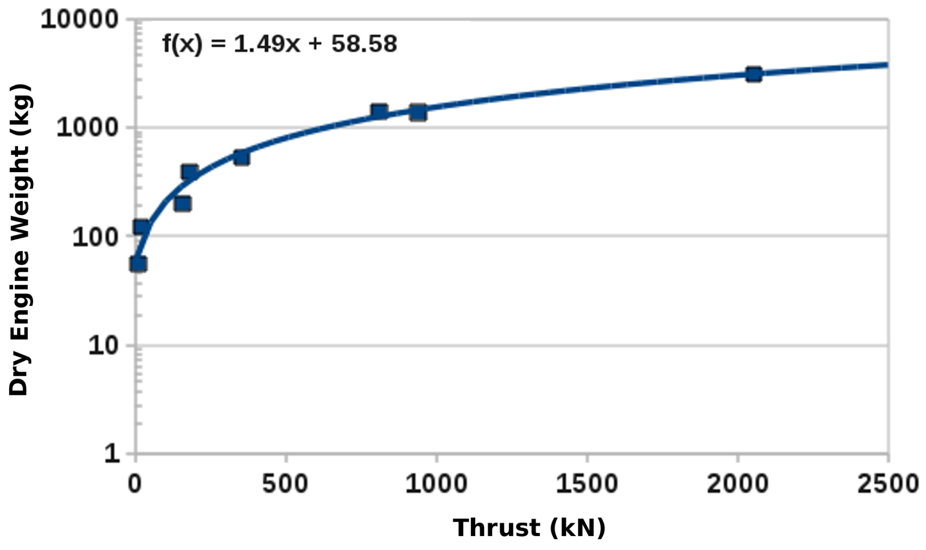


Figure 81: Sizing Fit for LOX/Methane Engines. Prototype results from [48].

size for the radials, lines and riser. The tensile strength of Nylon fabric and F_{design}, from Equation 57, is used to determine the thickness required for the parachute fabric. After the parachute is sized, a mortar is sized by $m_{mortar} = 1.48\, m_{para}^{0.5}$ which was derived from historical mortar data.[49] PESST currently assumes that a mortar is used for the main parachute though larger parachutes typically have a drogue chute instead, released by a smaller mortar.

5.2.6.4 Backshell Sizing

$$m_{backshell} = 6.7582\, m_{entry}^{0.4116} \tag{58}$$

The backshell mass is estimated by Equation 58 and assumes that the backshell is integrated with the structure of the vehicle. For smaller vehicles, where the regression would extrapolate, the mass fraction for the backshell and primary structural mass is capped at 25%.

5.3 Comparison to a Historical Mission

PESST predictions for a historical mission is included to display the suitability and flexibility of the framework for conceptual design.

Table 20: Trajectory and Heating Comparison to Mars Pathfinder.

Event	Units	Flight Reconstruction [19, 64]	POST	PESST	Diff from POST (%)	Diff from Reconst. (%)
Initial Conditions						
Time from Entry	s	0	0	0	-	-
Altitude	km	128.0	128.0	128.0	-	-
Relative Velocity	m/s	7479.0	7479.0	7479.0	-	-
Flight Path Angle	deg	-13.65	-13.65	-13.65	-	-
Peak Heat Rate Conv (Rad)	W/cm^2	118.0 (5.5)	115.5 (-)	115.3 (4.8)	-0.2 (-)	-2.3 (-12.4)
Time from Entry	s	66.0	62.3	62.2	-0.2	-5.8
Altitude	km	42.3	40.4	40.8	0.9	-3.7
Relative Velocity	m/s	6589.6	6512.9	6552.2	0.6	-0.6
Heatload Conv (Rad)	J/cm^2	3800.0 (88.7)	4335.0 (-)	4330.7 (84.2)	-0.1 (-)	14.0 (-5.0)
Peak Deceleration	Earth-g	15.8	16.3	16.3	-0.3	2.7
Time from Entry	s	77.8	72.9	73.4	0.7	-5.6
Altitude	km	33.1	31.8	31.8	-0.1	-4.1
Relative Velocity	m/s	5055.3	5032.9	5029.2	-0.1	-0.5
Parachute Full Inflation	Earth-g	6.0	7.5	6.0	-20.3	0.0
Time from Entry	s	173.2	161.3	163.0	1.1	-5.9
Altitude	km	7.6	8.2	8.2	-0.4	7.4

Table 20: Trajectory and Heating Comparison to Mars Pathfinder.

Event	Units	Flight Reconstruction [19, 64]	POST	PESST	Diff from POST (%)	Diff from Reconst. (%)
Relative Velocity	m/s	338.5	369.8	360.6	-2.5	6.5
Heatshield Drop						
Time from Entry	s	192.2	182.6	192.0	5.1	-0.1
Altitude	km	6.9	6.5	5.9	-9.7	-15.0
Relative Velocity	m/s	94.3	82.2	79.0	-3.8	-16.2
Trajectory Termination						
Time from Entry	s	298.0	290.0	285.2	-1.7	-4.3
Altitude	m	130.7	128.6	127.4	-0.9	-2.5
Relative Velocity	m/s	63.3	53.7	56.9	6.0	-10.1

The reconstruction for Mars Pathfinder comes from the work of Christian, et al. [19] which created a Pathfinder reconstruction using an extended Kalman filter. Heating information is taken from Milos, et al. [64]. The reconstructed Pathfinder atmosphere is used for both the PESST and POST trajectory. Both PESST and POST specify the same dynamic pressure trigger for parachute inflation and the same aerodynamics for the Disk-Gap Band parachute. Differences in the implementation of the infinite mass parachute inflation model may explain the discrepancy between the two tools. Otherwise, all three show good agreement for the purposes of conceptual design, Tables 20 and 21.

Table 21: Mass Comparison to Pathfinder.

Element (kg)	**Flight Mass**	**PESST**	**Diff Flight** (%)	**Flight Mass** (%)
Entry Mass	585.3	585.0	-0.1	-
Heatshield	64.4	75.0	16.5	11.0
Backshell and Structure	56.9	93.0	63.4	9.7
Parachute	9.8	17.0	74.4	1.7
Payload[1]	360.0	360.0	0.0	61.5
Remainder	94.2	40.0	-57.6	16.1

Any conceptual design tool needs to predict the full system with enough accuracy to inform initial design decisions. Martian mission designers can be used to designing probes to withstand 20 g_E but PESST can allow them to better understand a new entry environment, such as Venus, where missions may experience 250 g_E. In this way, the framework is broadly applicable to better understanding EDL systems. The validation case for Pathfinder shows that this is a realistic conceptual design tool for entry vehicles. Using the tool as an example for link importance ranking will assist in validating the method by grounding the results to realistic design cases.

[1]Specified by user

5.4 Summary

Conceptual design decisions are better informed by the early application of physics-based tools to the design process. First order tools are often fast enough to be used during parametric studies while still providing information on many of the main effects and trades that can now be examined, far earlier in the design process, for entry missions. PESST fills an important place in the field of entry design synthesis by allowing a designer the chance to quickly gain an understanding of a design space by using first-order physical models. A detailed examination was made into the methods used to implement each of the entry discipline modules, allowing for the widespread use and critique of these models for entry systems.

CHAPTER VI

SUMMARY AND RECOMMENDATIONS

6.1 Summary

Though the scale of feasible multidisciplinary problems has been greatly increasing over the past twenty years, improvements in processing power and decomposition methods will be required to push even larger problems into feasibility. Further improvement to the currently utilized decomposition methods are required to efficiently solve complex engineering problems. This investigation advances the state-of-the-art in importance heuristics, static decomposition and EDL system synthesis.

6.1.1 Introduction of a Link Importance Heuristic

Mutual information was introduced as a useful new importance heuristic for ranking link variables. The metric was shown to measure both linear and non-linear dependence between variables. Mutual information was compared to the standard use of correlation by applying both to a realistic conceptual design study. The Planetary Entry Systems Synthesis Tool (PESST) was utilized to perform this design study on a Mars entry. A design space of entry conditions and vehicle geometries was explored and trajectory variables were made available to ranking at set altitudes.

The metric was able to correctly determine that total propulsion mass was just as important as total propulsion system mass as both are strong indicators for the triggering of the guidance event. Linear correlation was unable to determine the high importance of total propulsion system mass while mutual information was able to correctly determine the most important variables.

This metric provides the ability to better understand the driving variables of a design study. With a better understanding of modeled behavior, synthesis studies can tell engineers more about the systems they model and more quickly expose weaknesses within these model. Here an example showed how mutual information could expose the importance of the triggering of the guidance event and the most important driving variables towards meeting a specified objective function.

The use of force-based clustering on the low thrust trajectory problem selected showed the value of mutual information for ranking potential cut points in long trajectory problems. Force-based clustering was able to discover useful sub-problem structure that led to a 20% reduction to the total run-time of the trajectory solver.

6.1.2 Introduction of a Global Optimizer for Static Decomposition

A robust comparison was performed between MIMIC and Genetic Algorithms for static decomposition. This comparison introduces MIMIC as an optimizer well suited for the static decomposition domain. Both methods are stochastic requiring a large number of runs to compare averaged behavior and requires a search over a specified setting domain to find proper settings for each method. A mixed Latin-Hypercube and 2-level Full Factorial design was created to search the domain of settings. Five randomly generated problems were created from six different configurations of difficulty. To run the cases within a reasonable amount of time, both methods were parallelized and run on a cluster of GPU processors. The recent advances in GPU computing were leveraged to deliver a comparison uncommon in the literature.

This comparison was enabled by new advances in parallel computing hardware. A comparison was made regarding the ease of parallel implementation for both methods. The largest shortcoming to the parallel implementation of MIMIC is remedied by the explanation of the method in this work and the information theoretic concepts involved. A time breakdown for each element of the method is provided to forewarn individuals who wish to try to develop new implementations and adjustments to the method. Areas to focus on for future implementations are suggested during the discussion on future work.

MIMIC was shown to strongly out perform Genetic Algorithms on the number of function calls required to reach a specified target value and was less prone to fall into local optima. The Genetic Algorithm was able to out perform MIMIC on the run-time required for loosely coupled problems (i.e. 5% active links). A run-time breakdown of the MIMIC algorithm showed that the greater run-time required by MIMIC for these problems came from a computational overhead that was not sensitive to the cost of a fitness function. The fitness function used in this comparison study took milliseconds to compute. Based on the averaged convergence behavior seen though this work, as the cost of the fitness function increased, MIMIC would perform with better run-time performance for even loosely coupled problems. Even with the inexpensive function call used in this study, the run-time performance is competitive with Genetic Algorithms over medium and strongly coupled problems due to MIMIC's speed of convergence though the efficient use of function calls. This makes the algorithm potentially applicable for problems with large function evaluation costs or for large highly coupled problem domains.

The performance of the MIMIC algorithm was also shown to be more robust than Genetic Algorithms when using settings within 20% of the best settings found for either optimizer. This is important as the ideal settings to use on a problem are never known in practice. Getting near the ideal setting for a Genetic Algorithm was shown to provide performance that was less predictable than when using near-best settings with MIMIC. Parzen-window density estimation was also shown to perform adequately to allow the method to also treat continuous variables.

Several published utility functions have been shown for reordering a problem and are normally applied by using a global optimizer. A global optimizer is generally used as this domain contains many local optima. The efficiency of MIMIC relaxes the computational constraints that have been present for researchers. The algorithm provides a marked advancement for researchers as a replacement to the standard practice of using Genetic Algorithms for utility function research.

6.1.3 Generalizing Static Results with Automatic Decision Trees

Machine learning algorithms were utilized to automatically generate decision trees to generalize results obtained during the static decomposition comparison between MIMIC and Genetic Algorithms. Useful descriptive metrics were developed that could predict when one optimizer would be better for a given decomposition problem. The large data set generated during the optimizer comparison was used to train the decision trees. The tree structure was pruned automatically to leave a well performing structure that was tested for accuracy through 10-point cross-validation.

With imperfect knowledge of the ideal settings, the most important predictor of which method to use was the degree of coupling present in the problem. All medium and strongly coupled problems were predicted as being better served by use of the MIMIC optimizer. The number of pair-wise coupled analyses (e.g. A $\rightarrow$ B and B $\rightarrow$ A) was also determined to be a useful descriptive metric to method performance. High pair-wise coupling of analyses was better handled by MIMIC while lower degrees of pair-wise coupling showed averaged performance that was too similar to determine which method to use within a 90% confidence. These results clarify the comparison charts and provide a user with descriptive metrics that could be used on other problems to determine if the problem might perform well using MIMIC, Genetic Algorithms, or if the best optimizer could not determined by this study to within a 90% confidence.

6.1.4 Creation of a Design Synthesis Tool for Entry Design

The Planetary Entry Systems Synthesis Tool (PESST) is a rapid conceptual design tool that was developed as part of this work for entry, descent, and landing systems. This framework has the capability to estimate the performance of an entry system using user-defined geometry, hypersonic aerodynamics, flight mechanics, thermal response, and mass estimation. Trade studies can be performed by parameter sweeps to gain a valuable understanding of the design space for conceptual studies. This framework is broadly applicable to conceptual studies of EDL systems and has been used as a test case for the evaluation of the proposed link importance metric. PESST fills an important place in the field of entry design synthesis by allowing a designer the chance to quickly gain an understanding of a design space by using first-order physical models.

6.2 Recommendations for Future Work

6.2.1 Static Decomposition

MIMIC has been compared here against Genetic Algorithms on a static decomposition problem in order to appropriately schedule the analyses on the DSM based on a utility function. The MIMIC algorithm may also have many other applications on problems normally solved by the use of a Genetic Algorithm. On multimodal domains with non-trivial function calls; MIMIC should be strongly considered as an optimizer. As an example, the method could be applied as an down-selection tool for globally searching a multimodal trajectory problem before the use of a local optimizer.

The MIMIC method is composed of three adaptable components: Model building, Sampling, and Evaluation. The modeling of the joint distribution is the key to the method. Any method could be used to model the joint distribution, MIMIC as it is implemented here forms the best possible model out of pair-wise relationships. Another model could be used instead that took advantage of more complex relations to yield a higher fidelity model of the joint distribution. For instance, instead of pair-wise Chow Lu trees to model discrete distributions one could use a t-cherry junction tree to approximate the true joint distribution using triplet interactions. T-cherry trees have been shown to reduce the KL-distance to true distributions when compared to

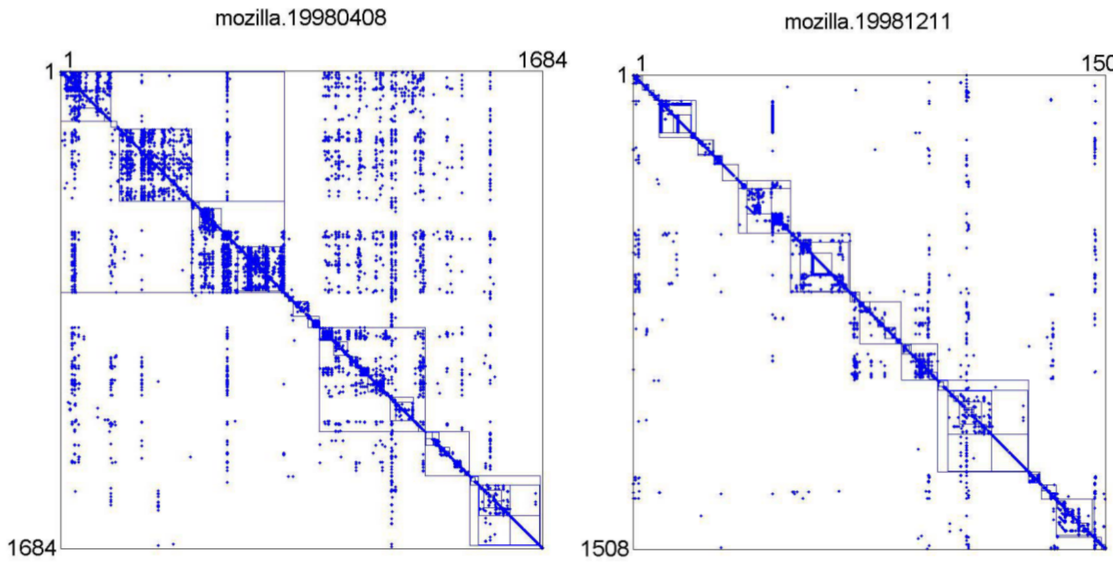
Chow Lu trees. Essentially, future work replacing Chow Lu trees with t-cherry trees would not hurt the approximation and might help it over more complicated joint distributions. Several potential drawbacks to the model replacement would need to be explored regarding the potential increase in computational complexity, the number of samples required to generate a model, and how the parallelizability of the addition compares to Chow Lu trees.

Sampling from the created model takes the majority of time in the current implementation (i.e. +70%). This percentage increases as the problem size increases and decreases as the function call cost increases. It was seen that when the averaged function call cost surpassed 11 ms, the cost of sampling did not prevent MIMIC from becoming the faster algorithm.

To lower this averaged crossover function call cost of 11 ms, the sampling method used by MIMIC could be adapted to more efficiently execute conditional statements. Conditional statements are not currently performed as efficiently on a GPU due to the lack of predictive scheduling that is normally utilized on general purpose CPUs. Tree structures contain many conditional statements that must be executed as the tree is traversed; i.e. take left branch if true else take right branch. Sampling involves the repeated traversal of the built tree model which is one factor to consider when designing a more efficient sampling component. The author predicts that GPUs will become more efficient in evaluating conditional statements just as CPUs have been incorporating the idea of having 'many-cores' into their framework. Further work on a GPU parallized sampling component will require a thorough understanding of GPU limitations and capabilities.

For instance, a warp is a basic thread group in the NVIDIA CUDA framework. If a warp of threads is able to travel through the same conditional choices together than all threads can execute in parallel. If one thread in the warp needs to travel through a different conditional choice, all threads in the warp are currently required to execute in series. If the model could be made into one that required fewer conditional choices or if formed warps were composed of threads likely to make the same selections; the time required for sampling can be drastically reduced when using a GPU framework.

One example of composing warps likely to make the same selections can be realized by sampling along the tree in a step-wise fashion with all threads initially computing the root node. At that point, threads could be grouped based on the path determined from the value generated at the root node. Each child node with a group of threads would have these threads executed in parallel and then grouped by their membership with lower-level nodes, based on the computed value.



(a) Mozilla DSM Before Redesign (b) Mozilla DSM After Redesign

Figure 82: DSM Orderings for the Mozilla Browser: Before and After Redesign Effort[59]

Future work could also examine the use of static decomposition as a guide for the modular redesign of large aerospace software applications. The modular design of software and systems aids in the maintenance, reuse and testing of the system. An earlier version of the Mozilla browser had a large number of interconnections between software components, Figure 82(a). A redesign effort was able to create a more stable modular design shown in Figure 82(b) with fewer connections between modules.[59]

There are many complex interconnected projects in Aerospace (e.g. guidance systems, computational fluid dynamics simulations, high fidelity heatshield sizing, etc.) that would benefit from the adapted use of static decomposition during a redesign effort.

6.2.2 Force-Based Clustering

The method developed in this work to first create pork-chop plots, transform them into probability distributions, and then calculate the mutual information for each transfer is promising but much more work is needed to mature the method. The work performed on one trajectory problem should be applied to a wide sweep of randomly created trajectory problems as was performed for the static decomposition comparison. The wide selection of problem would help determine which types of trajectory problems are best addressed by mutual information link ranking.

When using pork-chop plots as a method for determining the mutual information of a trajectory link, a great number of variables need to be further examined. The time period that the plots cover should be examined. The plots could run over one synotic period or could each span a year with the assumption that the vehicle would likely meet a target once a year. The plots could be pre-generated or run for a set period from the exact date the vehicle leaves one body. Users with the capability to utilize massively parallel computational hardware such as graphics cards would be aided by the decomposable nature of the pork-chop plot computation. Ballistic pork-chop plots will not be applicable to all trajectories. Other types of plots could be used in the same way to determine an importance rank for each trajectory linkage.

When the attractive force of edges are proportional to the mutual information weight assigned to the edge, clusters will form between analyses that share a large number of high quality links. Analyses with few low importance links or those with one mid-importance link will be driven further from the discovered clusters. The first links to consider for removal during dynamic decomposition could be the longest links in the graph while the links that are desirable to keep, for clustering, are the shortest links.

If the process of graph link removal is to be automated for future work with dynamic decomposition, a cluster quality metric is required to judge when removing links is no longer increasing the quality of the remaining clusters. Cluster quality metrics remain an area of active research but results to date generally demonstrate the benefit of forming closely coupled groups of analyses with few interconnections between analyses.[83] Force-based clustering can be a flexible method for future work on dynamic decomposition.

6.2.3 EDL Design Synthesis

The PESST framework is composed of many discipline modules that each possess a different level of fidelity. While all are conceptual design tools there is a definite difference between mass estimating relationships and 1-D thermal conduction equations. The 1-D conduction equations are dependent on a correct description of the heating environment from stagnation point heating approximations. It is suggested that any framework involving hybrid fidelity components should have some way of tracking that fidelity and how each component's variance could effect a system wide analysis. This would be valuable towards forming a prioritized list of which components should be refined in order to best increase the accuracy of system calculations.

PESST development would benefit from added functionality to increase module flexibility. Currently, if modules did not exist within the tool, a user would need to contact the maintainer to have them create the needed functionality. The usefulness of a conceptual design framework would be greatly benefited by decentralizing the task of module development. Users should be able to develop and plug-in their own modules to those existing in the framework, potentially allowing for the developing of a community supported EDL module library.

Another advancement towards the future development of the framework would be the addition of an automated component and system testing framework. Component tests are essential and components should fall within an acceptable margin from higher fidelity tools. A testing suite could be automatically run when changes are performed to confirm that all tests have been passed. The initial set up time would be well repaid by frequent automated testing that would increase robustness going forward as components are continually adapted into the future.

The most time intensive component in PESST is the TPS sizer. The time taken by this component is directly proportional to the number of trajectory data points passed through the thermal input file. The current component assumes equidistant time-steps in the conduction equations which means that the number of total time-steps is driven by the size of time-step needed to describe the most variable region of the trajectory. A short Martian entry can use 0.01 second time-steps but a Venus mission normally experiences its heating spike during the first minute of an hour long entry. This either provides a large number of needless points to the TPS sizer, greatly increasing its run-time, or drives the developer to set an arbitrary limit as to only use the first 2000 time-steps. The setting of a time-step limit potentially cuts off the heating pulse of a skip entry where the second entry could occur after the time-step cutoff. The real solution is frame the CFD problem so the step sizes are equal in the domain of computation but unequal physically. The points passed to the sizer would then be based on a percentage rule. For example, only passing points that are a set percentage of change from the highest value seen so far. This would drastically reduce the time taken to compute the TPS thickness; potentially cutting the total time of execution by more than half. It would also increase the robustness of the code for skipping entries by allowing the consideration of heating over the entire trajectory. The current PESST framework only examines the first 2000 time-steps for the heating environment. This is suitable for most missions but might not be suitable for skipping trajectories.

The aerodynamics tool (AeroMesh) was developed for PESST and was created as a separate executable for stand alone use. It can calculate hypersonic aerodynamic coefficients given a provided vehicle geometry. By incorporating the AeroMesh module into the main PESST executable, trade studies could be performed over ranges of vehicle geometry. A first draft of a PESST version that does this is available from the main development branch of the PESST repository but requires further testing and refinement before public release.

Currently the PESST framework can fly a given shape at a specified angle of attack but no analysis is currently performed as to whether the angle of attack is stable. An adaptation to the current AeroMesh program would be able to analyze the trim-line for the specified trim angle of attack (i.e. where $C_M = 0$) and compute the derivative of C_M with respect to angle of attack.

APPENDIX A

THERMAL DATA UTILIZED BY PESST

Useful tables of public domain TPS material information, that are both unclassified and approved for universal release, have been included for the ease of the reader. This material property data is utilized by PESST when calculating the amount of TPS material required to insulate the vehicle from entry heating. The factors used to covert the original Imperial units to SI have been included in Table 22.

Table 22: Conversion Factors

SI	**Imperial**
1 kg/m3	6.242782×10^{-2} lbm/ft3
1 kJ/kg-K	2.388459×10^{-1} Btu/lbm-R
1 W/m-k	1.604969×10^{-4} Btu/ft-s-R
1 kJ/kg	4.299208×10^{-1} Btu/lbm

Table 23: PICA Density and Reaction Information. [96]

Component	**Initial Density**	**Residual Density**	**Pre-exponential Factor**	**Density Exponent**	**Activation Energy**	**Min Reaction Temp**
	(kg/m^3)	(kg/m^3)	(s^{-1})		(K)	(K)
Resin (Comp A)	228.26	0.00	1.400E+04	3.0	8555.56	555.56
Resin (Comp B)	973.12	792.92	4.480E+09	3.0	20444.45	333.33
Fiber Reinforcement	160.18	160.18	0.000E+00	0.0	0.00	50000.00

Table 24: PICA Heats of Formation and Resin Fraction. [96]

Resin Volume Fraction	**Virgin Heat of Formation** (kJ/kg)	**Char Heat of Formation** (kJ/kg)	**Pyrolysis Gas Heat of Formation** (kJ/kg)	**Datum Temp for Zero Enthalpy** (K)
0.0646	-875.7	0.0	0.0	297.8

Table 25: PICA Virgin Properties for Insulation. [96]

Temp (K)	**Cp** (kJ/kgK)	**k** (W/mK)	ϵ	α
222.2	1.44863	5.48584E-01	0.80	0.80
527.8	1.93430	5.85968E-01	0.80	0.80
833.4	2.34042	6.25887E-01	0.80	0.80
1650.6	2.46603	1.08696E+00	0.80	0.80
2467.8	2.46603	1.54366E+00	0.80	0.80
2608.2	2.46603	1.56327E+00	0.80	0.80
2748.6	2.46603	1.60968E+00	0.80	0.80
2888.9	2.46603	1.64362E+00	0.80	0.80
3333.3	2.46603	1.68100E+00	0.80	0.80

Table 26: PICA Char Properties for Insulation. [96]

Temp (K)	**Cp** (kJ/kgK)	**k** (W/mK)	ϵ	α
294.4	2.93076	9.78212E-01	0.90	0.90
425.0	3.76812	9.90673E-01	0.90	0.90
555.6	4.60548	1.00313E+00	0.90	0.90
833.3	4.60548	1.17136E+00	0.90	0.90
1666.7	4.60548	1.33959E+00	0.90	0.90
3333.3	4.60548	1.71343E+00	0.90	0.90

Table 27: PICA Pyrolysis Gas Entropy. [96]

Temp (K)	**Enthalpy** (kJ/kg)
400	-9.0172E+03
800	-6.1716E+03
1200	-2.2682E+02
1600	1.2874E+03
2000	2.8954E+03
2400	5.2126E+03
2800	9.7202E+03
3200	1.9148E+04
3600	3.3278E+04
4000	4.4557E+04
4400	5.0000E+04
4800	5.3161E+04

Table 28: FM 5055 Density and Reaction Information. [45]

Component	**Initial Density**	**Residual Density**	**Pre-exponential Factor**	**Density Exponent**	**Activation Energy**	**Min Reaction Temp**
	(kg/m^3)	(kg/m^3)	(s^{-1})		(K)	(K)
Resin (Comp A)	247.97	0.00	2.171E+01	1.92	4627.22	297.78
Resin (Comp B)	0.00	0.00	1.000E+00	1.00	0.56	5500.00
Fiber Reinforcement	2231.06	1905.88	9.535E+10	3.10	19333.33	560.56

Table 29: FM 5055 Heats of Formation and Resin Fraction. [45]

Resin Volume Fraction	**Virgin Heat of Formation**	**Char Heat of Formation**	**Pyrolysis Gas Heat of Formation**	**Datum Temp for Zero Enthalpy**
	(kJ/kg)	(kJ/kg)	(kJ/kg)	(K)
0.4010	-911.1	0.0	0.0	297.8

Table 30: FM 5055 Virgin Properties for Insulation. [45]

Temp	**Cp**	**k (Across Ply)**	ϵ **(est)**	α **(est)**
(K)	(kJ/kgK)	(W/mK)		
311.1	0.96296		0.85	0.85
338.9		8.09984E-01	0.85	0.85
366.7	1.19324		0.85	0.85
422.2	1.29791	9.22136E-01	0.85	0.85
477.8	1.37327	9.78212E-01	0.85	0.85
533.3	1.44445	1.02183E+00	0.85	0.85
588.9	1.48631	1.04052E+00	0.85	0.85
644.4		1.00937E+00	0.85	0.85

Table 31: FM 5055 Char Properties for Insulation. [45]

Temp	**Cp**	**k (Across Ply)**	ϵ **(est)**	α **(est)**
(K)	(kJ/kgK)	(W/mK)		
311.1	0.85829		0.85	0.85
366.7	1.00483	9.53289E-01	0.85	0.85
422.2	1.13881		0.85	0.85
477.8	1.23511	1.17136E+00	0.85	0.85
533.3	1.33978		0.85	0.85
588.9	1.44445	1.35828E+00	0.85	0.85
644.4	1.50725		0.85	0.85
700.0	1.57005	1.51405E+00	0.85	0.85
755.6	1.62448		0.85	0.85
811.1	1.67472	1.64489E+00	0.85	0.85
866.7	1.71659		0.85	0.85
922.2	1.75846	1.67604E+00	0.85	0.85
977.8	1.77520		0.85	0.85
1033.3	1.79614		0.85	0.85
1088.9	1.80870	1.67604E+00	0.85	0.85

Table 32: FM 5055 Pyrolysis Gas Entropy. [45]

Temp (K)	**Enthalpy** (kJ/kg)
294.44	-3661.6
555.56	-2734.0
833.33	-1547.0
1111.1	-153.5
1200.0	351.2
1388.9	1093.2
1666.7	2046.9
1944.4	2930.8
2250.0	3981.4
2500.0	5036.0
2750.0	6298.1
3000.0	7944.0
3250.0	10186.1
3500.0	13254.8
3750.0	17358.5
4055.6	22602.8

APPENDIX B

STATIC PROBLEMS EXAMINED WITH BEST SETTINGS AND NEAR-BEST RESULTS

Table 33: Summary of DSM Problems Used for Static Testing

Number of Elements	**% of Links Active**	**DSM Problems Generated with these Characteristics**
15	5	5
15	10	5
15	20	5
25	5	5
25	10	5
25	20	5
50	5	5
50	10	5
50	20	5
	Total Problems	45

Table 34: Confidence Needed to Select Between Methods (Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
46	15	5	UNK	MIMIC	GA	50.2	99.9+
47	15	5	UNK	MIMIC	GA	95.1	99.9+
48	15	5	UNK	MIMIC	GA	98.4	99.9+
49	15	5	UNK	MIMIC	GA	82.2	99.9+
50	15	5	UNK	MIMIC	GA	91.2	99.9+
51	15	10	UNK	MIMIC	GA	98.9	99.9+
52	15	10	UNK	MIMIC	GA	21.8	99.9+
53	15	10	UNK	MIMIC	GA	82.8	99.9+
54	15	10	UNK	MIMIC	GA	44.3	99.8
55	15	10	MIMIC	MIMIC	MIMIC	97.9	84.1
56	15	20	MIMIC	MIMIC	GA	92.1	21.7
57	15	20	MIMIC	MIMIC	GA	92.4	36.9
58	15	20	UNK	MIMIC	GA	76.6	54.2
59	15	20	UNK	GA	GA	74.4	99.9+
60	15	20	UNK	MIMIC	GA	92.8	99.9+
0	25	5	UNK	GA	GA	99.9+	99.9+

Table 34: Confidence Needed to Select Between Methods (Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
9	25	5	UNK	GA	GA	99.9+	99.9+
18	25	5	UNK	GA	GA	99.9+	99.9+
27	25	5	MIMIC	MIMIC	MIMIC	99.6	93.1
36	25	5	UNK	GA	GA	99.9+	99.9+
3	25	10	MIMIC	MIMIC	MIMIC	98.8	45.2
12	25	10	UNK	GA	GA	99.9+	99.9+
21	25	10	MIMIC	MIMIC	GA	99.6	2.3
30	25	10	MIMIC	MIMIC	MIMIC	99.9+	99.9+
39	25	10	MIMIC	MIMIC	MIMIC	94.4	69.8
6	25	20	MIMIC	MIMIC	MIMIC	98.3	39.0
15	25	20	UNK	MIMIC	GA	99.5	99.9+
24	25	20	MIMIC	MIMIC	MIMIC	98.1	83.1
33	25	20	UNK	MIMIC	GA	25.8	99.9+
42	25	20	UNK	MIMIC	GA	87.1	65.7
1	50	5	UNK	GA	GA	99.9+	99.9+
10	50	5	MIMIC	MIMIC	GA	99.9+	99.9+

Table 34: Confidence Needed to Select Between Methods (Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
19	50	5	MIMIC	MIMIC	GA	85.2	94.6
28	50	5	MIMIC	MIMIC	GA	43.0	99.9+
37	50	5	MIMIC	MIMIC	GA	57.1	99.7
4	50	10	MIMIC	MIMIC	GA	99.9+	70.1
13	50	10	MIMIC	MIMIC	GA	99.9+	99.9+
22	50	10	UNK	GA	GA	99.9+	99.9+
31	50	10	MIMIC	MIMIC	MIMIC	99.9+	97.7
40	50	10	MIMIC	MIMIC	MIMIC	97.2	98.9
7	50	20	UNK	GA	GA	99.9+	99.9+
16	50	20	MIMIC	MIMIC	GA	68.4	99.9+
25	50	20	MIMIC	MIMIC	MIMIC	99.9+	17.9
34	50	20	MIMIC	MIMIC	GA	99.9+	99.9+
43	50	20	MIMIC	MIMIC	GA	97.2	99.9+

Table 35: Confidence Needed to Select Between Methods (Near-Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Convergence Choice %	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
46	15	5	UNK	MIMIC	GA	99.9+	99.9+	99.9+
47	15	5	UNK	MIMIC	GA	99.9+	99.5	99.9+
48	15	5	UNK	MIMIC	GA	99.9+	99.7	99.9+
49	15	5	UNK	MIMIC	GA	99.9+	93.4	99.9+
50	15	5	UNK	MIMIC	GA	99.9+	99.7	99.9+
51	15	10	MIMIC	MIMIC	GA	25.8	15.5	99.9+
52	15	10	MIMIC	MIMIC	GA	93.4	98.8	76.4
53	15	10	UNK	GA	GA	99.9+	56.4	89.4
54	15	10	GA	MIMIC	GA	45.4	11.0	91.4
55	15	10	MIMIC	MIMIC	GA	99.9+	99.9+	90.0
56	15	20	MIMIC	MIMIC	GA	99.9+	99.9+	92.8
57	15	20	MIMIC	MIMIC	GA	99.9+	99.9+	96.2
58	15	20	MIMIC	MIMIC	GA	94.1	98.6	99.5
59	15	20	MIMIC	MIMIC	GA	63.4	68.3	97.1
60	15	20	MIMIC	MIMIC	GA	99.7	99.9+	97.6

Table 35: Confidence Needed to Select Between Methods (Near-Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Convergence Choice %	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
0	25	5	UNK	GA	GA	99.9+	99.9+	99.9+
9	25	5	UNK	GA	GA	99.9+	91.4	99.9+
18	25	5	MIMIC	MIMIC	GA	86.9	91.7	92.8
27	25	5	MIMIC	MIMIC	GA	99.9+	99.9+	99.5
36	25	5	UNK	MIMIC	GA	99.9+	64.1	81.3
3	25	10	MIMIC	MIMIC	GA	99.9+	99.9+	94.5
12	25	10	UNK	MIMIC	GA	99.9+	44.5	99.9+
21	25	10	MIMIC	MIMIC	GA	99.9+	99.9+	51.1
30	25	10	MIMIC	MIMIC	MIMIC	99.9+	99.9+	36.5
39	25	10	MIMIC	MIMIC	MIMIC	99.9+	99.9+	97.6
6	25	20	MIMIC	MIMIC	GA	95.8	98.6	84.8
15	25	20	MIMIC	MIMIC	GA	99.8	99.9+	94.1
24	25	20	MIMIC	MIMIC	GA	99.4	99.9+	80.3
33	25	20	MIMIC	MIMIC	GA	88.9	94.5	89.9
42	25	20	MIMIC	MIMIC	GA	16.1	85.4	99.0

Table 35: Confidence Needed to Select Between Methods (Near-Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Convergence Choice %	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
1	50	5	UNK	GA	GA	99.9+	99.9+	99.9+
10	50	5	MIMIC	MIMIC	GA	99.9+	99.9+	99.7
19	50	5	MIMIC	MIMIC	GA	99.6	99.8	99.9+
28	50	5	MIMIC	MIMIC	GA	98.7	22.5	99.9+
37	50	5	MIMIC	MIMIC	GA	98.9	97.5	99.9+
4	50	10	MIMIC	MIMIC	GA	99.9+	99.9+	70.5
13	50	10	MIMIC	MIMIC	GA	99.9+	99.9+	99.9+
22	50	10	MIMIC	MIMIC	GA	84.3	26.5	97.3
31	50	10	MIMIC	MIMIC	GA	99.9+	99.9+	87.6
40	50	10	MIMIC	MIMIC	GA	99.3	99.9+	97.1
7	50	20	UNK	GA	GA	99.9+	7.4	32.7
16	50	20	MIMIC	MIMIC	GA	99.9+	85.9	99.9+
25	50	20	MIMIC	MIMIC	MIMIC	99.9+	99.9+	3.4
34	50	20	MIMIC	MIMIC	GA	99.9+	91.2	36.5

Table 35: Confidence Needed to Select Between Methods (Near-Best Settings)

ProbID	Size (inputs)	Links Active %	Best for Convergence	Best for Function Calls	Best for Run-time	Confidence Needed for Convergence Choice %	Confidence Needed for Function Call Choice %	Confidence Needed for Run-time Choice %
43	50	20	MIMIC	MIMIC	MIMIC	99.9+	99.9+	12.1

Table 36: Best Settings found for Problems within Searched Setting Domain

ProbID	Size (inputs)	Links Active %	MIMIC Candidates	MIMIC Percent Modeled %	MIMIC Percent Noise %	GA Population	GA Percent Crossover %	GA Percent Mutation %
46	15	5	100	63.0645	7.45161	100	100.00000	15.00000
47	15	5	100	65	1	100	95.16130	7.58065
48	15	5	100	65	1	100	100.00000	5.00000
49	15	5	100	65	1	100	100.00000	15.00000
50	15	5	100	65	11	100	95.16130	7.58065
0	25	5	1000	50	5	100	70	15

Table 36: Best Settings found for Problems within Searched Setting Domain

ProbID	Size (inputs)	Links Active %	MIMIC Candidates	MIMIC Percent Modeled %	MIMIC Percent Noise %	GA Population	GA Percent Crossover %	GA Percent Mutation %
9	25	5	1000	35	1	140	93.7584	13.292
18	25	5	1000	35	1	267	85.1832	7.23748
27	25	5	1000	35	1	294	83.5484	11.4516
36	25	5	1000	65	1	100	100	5
1	50	5	1000	35	1	300	100	15
10	50	5	2742	49.5161	1.96774	300	100	15
19	50	5	1452	53.3871	1	229	92.2581	9.19355
28	50	5	2935	44.6774	4.87097	184	89.3548	12.4194
37	50	5	2161	47.5806	3.58065	100	100	15
51	15	10	229	54.3548	4.54839	300	100.00000	15.00000
52	15	10	294	41.7742	9.70968	300	100.00000	15.00000
53	15	10	100	35	11	133	87.41940	14.67740
54	15	10	113	46.9417	7.54668	268	100.00000	7.90323
55	15	10	616	45.6452	3.25806	176	91.21400	10.11930
3	25	10	1036	44.0935	7.60683	294	83.5484	11.4516
12	25	10	1000	35	1	184	89.3548	12.4194

Table 36: Best Settings found for Problems within Searched Setting Domain

ProbID	Size (inputs)	Links Active %	MIMIC Candidates	MIMIC Percent Modeled %	MIMIC Percent Noise %	GA Population	GA Percent Crossover %	GA Percent Mutation %
21	25	10	1000	50	5	274	73.871	9.83871
30	25	10	1323	64.0323	2.93548	197	77.7419	15
39	25	10	1013	46.9417	7.54668	300	70	15
4	50	10	2613	60.1613	2.29032	113	72.9032	13.0645
13	50	10	1516	45.6452	3.25806	267	85.1832	7.23748
22	50	10	1000	65	1	300	100	15
31	50	10	2613	60.1613	2.29032	242	86.4516	14.0323
40	50	10	2484	38.871	1.64516	300	93.2258	14.3548
56	15	20	745	48.5484	6.16129	300	93.22580	14.35480
57	15	20	874	56.2903	3.90323	300	100.00000	15.00000
58	15	20	810	59.1936	7.12903	261	76.77420	13.38710
59	15	20	294	42	10	235	97.09680	13.70970
60	15	20	681	36.9355	1.32258	300	70.00000	15.00000
6	25	20	1516	45.6452	3.25806	300	100	15
15	25	20	1323	64.0323	2.93548	294	83.5484	11.4516
24	25	20	1452	53.3871	1	242	86.4516	14.0323

Table 36: Best Settings found for Problems within Searched Setting Domain

ProbID	Size (inputs)	Links Active %	MIMIC Candidates	MIMIC Percent Modeled %	MIMIC Percent Noise %	GA Population	GA Percent Crossover %	GA Percent Mutation %
33	25	20	1000	50	5	300	93.2258	14.3548
42	25	20	1323	64.0323	2.93548	300	100	15
7	50	20	1581	36.9355	1.32258	300	93.2258	14.3548
16	50	20	2871	55.3226	5.19355	300	70	15
25	50	20	3000	35	1	300	100	15
34	50	20	2511	37.0328	6.64588	300	100	15
43	50	20	2613	60.1613	2.29032	300	70	15

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
46	15	5	0	0	140.7	20.66	155	27.92	10.58	1.907	1.798	0.3239

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
46			0	0	153	19.58	222	41.2	10.4	2.188	2	0.3711
46			0	0	174	21.36	204	32.34	14.94	2.8	1.955	0.31
46			0	0	204	29	222	41.2	12.65	1.798	1.927	0.3575
46			0	0	180	22.64	258	43.61	9.253	1.164	2.077	0.3512
46			0	0	152	30.04	160	30.19	23.29	6.247	1.988	0.3751
46			0	0	126	21.49	196	51.83	16.89	4.221	2.196	0.5807
46			0	0	160	19.09	232	54.76	15.06	1.798	2.347	0.554
46			0	0	140	18.79	212	30.6	12.39	1.663	2.232	0.3221
47			0	0	156	23.03	215	43.48	13.03	2.381	2.08	0.4207
47			0	0	219.9	34.87	276	60.93	29.41	6.664	2.217	0.4894
47			0	0	174	24.96	270	45	17.1	3.952	2.192	0.3653
47			0	0	265	41.24	288	48.35	19.9	3.145	2.224	0.3735
47			0	0	218.6	31.79	300	63.24	14.8	2.193	2.282	0.4811
47			0	0	161	25.73	224	40.05	28.43	6.252	2.308	0.4127
47			0	0	146.6	23.25	200	36.35	24.62	5.578	2.209	0.4015
47			0	0	173.6	30.31	216	35.85	17.26	3.049	2.207	0.3662

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
47			0	0	181.4	28.08	184	42.8	18.23	2.854	2.065	0.4803
48			0	0	261	42.68	370	60.97	33.63	6.214	2.823	0.4651
48			0	0	338.7	64.92	438	62.87	55.63	13.24	2.859	0.4104
48			0	0	284.7	30.55	462	85.06	45.4	6.342	2.975	0.5477
48			0	0	276.6	46.48	330	83.35	20.19	3.443	2.35	0.5936
48			0	0	299.8	32.39	402	74.79	22.91	2.508	2.666	0.496
48			0	0	216.8	33.44	296	40.62	48.49	9.385	2.729	0.3744
48			0	0	252.8	36.35	320	62.61	56.22	9.788	2.868	0.5611
48			0	0	317.9	87.06	356	68.5	37.88	10.44	3.035	0.5839
48			0	0	247.7	47.72	728	425.1	30.01	5.828	4.81	2.809
49			0	0	278.5	46.12	355	81.36	35.87	6.656	2.786	0.6385
49			0	0	352.2	93.41	1056	818.1	59.86	19.54	5.28	4.09
49			0	0	346.8	60.32	672	332.9	59.98	12.88	3.805	1.885
49			0	0	520.2	239.6	828	395.3	46.77	21.7	4.271	2.039
49			0	0	305.6	46.74	1800	1694	24.48	3.794	7.758	7.302
49			0	0	279.8	43.28	1080	1081	67.23	12.34	6.842	6.849

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
49			0	0	391.4	87.55	352	106.5	104.6	26.36	3.069	0.9283
49			0	0	571.4	377.4	1912	1817	70.78	46.92	10.66	10.13
49			0	0	727.4	536	6768	9321	92.95	68.68	33.63	46.31
50			0	0	278.5	45.35	365	69.87	37.05	6.762	2.79	0.534
50			0	0	371.1	55.91	336	63.4	66.9	12.26	2.475	0.4671
50			0	0	319.8	42.8	396	60.93	53.79	9.069	2.684	0.4129
50			0	0	305.6	46.74	618	59.52	24.02	3.723	3.496	0.3367
50			0	0	386.8	46.12	546	94.43	32.73	3.943	3.187	0.5512
50			0	0	303.2	47.96	328	44.68	80.87	14.96	2.895	0.3944
50			0	0	272.6	39.21	348	55.89	65.34	11.21	3.044	0.4889
50			0	0	271.1	37.78	388	55.89	32.51	4.565	3.153	0.4542
50			0	0	275	36.65	436	231.8	32.63	4.38	3.31	1.76
51	15	10	0	0	3085	351.2	6270	2030	225.2	25.8	15.26	4.941
51			0	0	20230	20880	74680	112300	2021	2093	140.5	211.3
51			0	0	78840	117100	18900	21290	7337	10910	37.26	41.98
51			0	0	18200	13560	14740	11850	1059	787	28.96	23.29

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
51			0	0	127200	138000	44230	47940	7044	7636	80.84	87.63
51			5	0	103600	164200	18140	11540	12440	19730	43.63	27.75
51			0	10	47240	45030	209300	225300	5991	5717	462.1	497.6
51			5	0	143500	163300	39410	37000	11890	13540	89.46	84
51			5	10	153900	162000	346300	239800	12100	12740	753.2	521.4
52			0	0	3614	1181	3825	430.9	161.6	52.09	10.24	1.153
52			0	0	3274	358.2	6354	3279	166	18.16	14.47	7.468
52			0	0	3592	453.6	68980	106600	183.5	23.17	130.5	201.7
52			0	0	4861	2459	90970	96290	176.4	87.12	169.2	179.1
52			0	0	4530	1353	16420	10980	160.6	46.76	31.87	21.32
52			0	0	5462	2221	3804	854.6	360.2	146.4	11.08	2.49
52			0	10	3350	950.8	253000	226100	217	61.56	573.8	512.8
52			0	5	4568	1667	106400	164000	212.6	76.27	241.2	371.8
52			0	5	13360	13010	106700	164000	634	613.9	235	361.4
53			0	0	964.5	137.3	1150	176.2	85.15	11.75	5.581	0.8549
53			0	0	1212	183.5	1272	164.8	101.8	15.16	5.621	0.7282

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
53			0	0	1114	207.9	1544	206.8	93.83	17.21	6.555	0.8778
53			0	0	1199	223.3	1520	155.7	77.63	13.83	6.162	0.6312
53			0	0	1007	145.2	1640	270	66.66	9.119	6.529	1.075
53			0	0	1058	202.7	1118	135.9	132.3	25.01	6.414	0.7797
53			0	0	1128	299.6	2109	726.4	140.3	36.8	10.23	3.523
53			0	0	1582	713.1	1431	276.1	157.3	69.32	7.328	1.414
53			0	0	17850	21080	2274	1529	1558	1835	10.33	6.947
54			0	0	2945	1485	3538	740.9	308.6	155.2	9.956	2.085
54			0	0	3448	1549	5522	1688	368.5	166.3	13.43	4.106
54			0	0	2104	458.6	61580	87740	225.2	49.46	121.8	173.6
54			0	0	4012	2461	18140	12380	315	191.6	36.84	25.15
54			0	0	4360	4182	5812	2123	325.5	309.8	13.45	4.911
54			0	5	4070	2989	180100	198100	638.7	470.1	432.6	475.7
54			0	0	9966	10900	31990	26070	2134	2336	78.83	64.22
54			5	0	102700	164300	6934	3408	13770	22030	18.57	9.127
54			5	0	150500	170900	3734	974.3	18340	20830	10.91	2.845

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
55			0	20	9728	2826	426000	297500	436.6	126.2	1108	774
55			0	30	11120	2350	737700	338400	640.1	136	1750	802.7
55			0	25	26970	26260	648300	332100	1595	1557	1504	770.6
55			0	35	28700	25310	760100	345600	1170	1024	1743	792.5
55			5	45	171100	186900	977500	352100	7575	8265	2194	790.4
55			5	35	106900	163900	854600	328200	6534	10030	2622	1007
55			5	50	116300	163300	1117000	354800	7088	9958	3359	1067
55			10	45	252700	222800	987400	348900	9951	8767	2944	1040
55			0	65	106900	137900	1329000	346300	4218	5438	3883	1012
56	15	20	0	10	11730	1799	249400	222700	597.9	91.51	518.8	463.3
56			0	20	14130	1049	408600	300300	953.3	71.49	784.1	576.3
56			0	25	12860	927.7	594600	317900	862.7	62.94	1112	594.7
56			0	30	14620	3883	728200	338000	674	176.5	1358	630.5
56			0	30	18620	11030	656600	334700	872.6	511.5	1201	612
56			0	30	58140	74630	750900	344900	8951	11510	2614	1201
56			5	35	120800	163200	749400	348600	12060	16310	1745	811.8

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
56			0	35	48450	59460	764200	346500	5490	6719	2274	1031
56			5	40	174700	179900	826600	362000	7906	8134	2481	1087
57			0	10	10830	1213	253200	224700	678.8	76.77	537.5	477.1
57			0	0	13390	1335	64440	54700	1484	152.1	130.9	111.1
57			0	20	19160	11070	527300	295800	1693	998	1739	975.9
57			0	20	24070	12300	436800	298300	2366	1203	839	572.9
57			5	10	130900	165100	269800	224900	9405	11850	500.8	417.4
57			0	40	9166	702.8	848800	355900	1610	126.9	2080	872.2
57			0	30	33090	24320	689700	337200	5051	3740	1640	801.8
57			0	40	15060	6898	829700	361800	1800	820.4	1983	864.7
57			5	35	169900	179300	788900	342800	10430	11000	1845	801.9
58			0	0	12100	2720	75470	87640	800.9	182.3	517.7	601.2
58			0	10	29310	24930	346700	243200	2849	2458	727.6	510.3
58			0	30	21020	11000	653800	337000	2037	1087	1338	689.5
58			5	10	121600	162900	212100	225100	7613	10200	430.9	457.2
58			5	30	134200	163500	718100	340300	8374	10200	1435	679.9

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
58			5	5	115400	163300	183500	185500	11490	16290	1007	1018
58			0	20	215300	140300	540800	305800	20530	13400	1410	797.4
58			5	10	143800	162900	285700	226500	7430	8415	747.7	592.9
58			30	15	610000	343600	540900	268500	31950	17990	1375	682.3
59			0	0	4835	946.5	4066	585	220.8	42.78	12.99	1.869
59			0	0	6212	705	26400	27900	341.5	38.75	60.39	63.84
59			0	0	5097	656.4	5132	1080	264.5	34.05	14.28	3.005
59			0	0	4648	497.2	52180	55510	279.9	29.19	110.9	118
59			5	5	105600	164000	108100	163900	4305	6681	226.7	343.7
59			0	5	4695	1093	109400	163900	319.7	74.42	305.8	458.4
59			0	0	53050	81270	8751	7538	5424	8308	26.91	23.18
59			5	0	103800	164200	11430	9125	7207	11390	34.17	27.28
59			0	15	23640	26740	310000	268100	1156	1303	849.4	734.5
60			0	0	7088	692.1	36160	26540	264.8	25.22	78.49	57.61
60			0	5	10100	1202	106600	163900	800.1	94.4	209.7	322.6
60			0	5	19650	18940	122700	163400	1286	1234	518.8	690.7

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
60			0	5	26850	24850	275800	220000	961.8	879.2	528.8	421.8
60			0	10	15100	9255	233900	224000	532.1	319	439.3	420.7
60			5	15	107200	163900	349100	263300	4826	7372	859.3	648
60			5	25	116300	163400	514100	323900	9631	13520	1522	959.2
60			5	10	135200	164800	287500	222300	7613	9260	684.3	529
60			5	20	113700	163400	525400	290100	4156	5964	1720	949.6
0	25	5	0	0	5050	404.3	1200	217.6	118	9.446	11.6	2.103
0			0	0	6190	905.7	1670	281.7	148	22.55	13.6	2.288
0			0	0	5420	822.3	1410	108.9	127	20.18	12	0.9255
0			0	0	6100	723.7	1720	258.6	118	13.46	13.3	2.003
0			0	0	6240	728	3460	2315	122	13.68	22.5	15.05
0			0	0	4960	450.8	1360	275.4	161	15.13	14.8	2.998
0			0	0	4640	384.4	4640	4192	150	12.85	40.1	36.29
0			0	0	4640	576.6	1680	404	113	13.51	16.6	3.995
0			0	0	3970	667.9	1500	251.9	92.7	15	15.1	2.542
9			0	0	5940	427.7	1540	138.8	126	8.635	12.2	1.097

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
9			0	0	8020	1204	1970	179.4	167	24.43	13.6	1.243
9			0	0	7880	822.1	2090	233.4	161	16.39	14.4	1.603
9			0	0	7770	568.5	2140	214	143	9.804	14	1.398
9			0	0	7250	635.6	2290	326.6	132	10.79	14.5	2.075
9			0	0	5810	407.1	2180	950.6	148	10.17	17.9	7.786
9			0	0	5810	566.6	15400	17550	149	14.19	97.4	111.1
9			0	0	5870	578.9	2000	434.8	130	12.1	16.4	3.57
9			0	0	5290	442.2	6180	4160	117	9.181	41.9	28.17
18			0	0	9060	1419	3930	385.1	202	30.57	18.1	1.773
18			0	0	10400	778.4	5090	865.4	218	16.01	21.4	3.643
18			0	5	10900	1564	107000	164000	232	32.6	311	476.2
18			0	0	11100	1042	10600	7278	211	18.97	36.2	24.9
18			0	0	9840	947.5	5520	501.8	188	17.14	21.4	1.944
18			0	0	7570	645.7	37800	30070	196	16.46	146	115.9
18			0	0	7850	726.8	18900	17350	203	18.52	74.5	68.3
18			0	0	6910	634.8	7950	4267	160	13.97	34	18.27

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
18			0	5	7140	760.6	106000	164000	164	16.62	371	576.6
27			0	30	31900	16130	627000	339900	741	370.8	2010	1088
27			20	35	438000	428900	1060000	346400	17800	17440	3130	1026
27			20	50	436000	429300	1030000	368400	18700	18450	2940	1052
27			20	35	440000	428800	832000	354300	20100	19540	2380	1013
27			20	45	495000	420900	967000	357400	10600	9034	4680	1729
27			30	40	697000	487500	954000	330000	20000	13970	3860	1334
27			40	40	893000	506400	1040000	337900	26300	14910	5210	1700
27			30	50	661000	482700	1160000	347500	55800	40690	4200	1261
27			40	50	1060000	507100	1070000	352100	87900	41930	3760	1234
36			0	0	3800	515	975	94.63	107	15.79	10.4	1.004
36			0	0	4900	289.5	1230	308.1	170	11.64	10.9	2.739
36			0	0	5320	283.7	1070	100.1	188	11.45	9.92	0.9304
36			0	0	5230	647.3	2360	1756	113	14.15	17.2	12.77
36			0	0	4770	524.7	99000	158400	96.9	10.76	524	838.8
36			0	0	3550	537.7	12100	17670	173	30.08	96.6	140.7

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
36			0	0	3510	572.9	3360	2921	171	32	29.5	25.65
36			0	0	3410	336.9	2420	2338	90.1	8.981	22.2	21.49
36			0	0	3640	500.1	3850	3006	416	57.68	32.9	25.73
3	25	10	0	15	41300	5246	430000	253300	1110	140.4	1420	836.3
3			0	20	57400	7072	583000	295100	1520	187.3	4180	2115
3			0	25	140000	139900	713000	316900	3770	3771	2100	934.5
3			0	50	177000	100300	1170000	327600	4000	2266	4680	1317
3			20	30	479000	423600	872000	312700	11100	9808	2480	888.6
3			0	35	260000	250000	783000	342700	9270	8927	3060	1340
3			10	60	231000	323400	1400000	322800	7300	10240	5280	1219
3			10	15	253000	320800	452000	272000	6740	8540	1710	1029
3			10	40	262000	319200	986000	348400	7500	9122	5650	1997
12			0	0	13000	945.7	4940	1097	299	21.3	30.6	6.802
12			0	0	15700	1192	5490	1020	350	26.32	31.4	5.826
12			0	0	14400	1376	5910	2478	323	30.42	32.6	13.65
12			0	0	15000	733.9	5770	608.3	302	14.26	31	3.268

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
12			0	0	13800	1042	14200	8259	276	20.05	62.9	36.55
12			0	0	10500	1407	90000	133000	287	37.89	502	741.3
12			0	0	10700	993.2	10500	6305	295	26.97	63.3	38.18
12			0	0	11200	747.4	13100	6825	268	17.37	77.5	40.34
12			0	0	11100	821.2	13800	7028	269	19.36	78.5	39.9
21			0	25	29400	3225	582000	313100	2100	230.1	2080	1120
21			0	20	110000	120200	604000	306100	3320	3648	2020	1023
21			0	45	77400	63570	1120000	345900	2180	1798	3530	1093
21			0	30	37100	5786	729000	342000	983	152.4	2360	1107
21			10	65	230000	323700	1330000	347400	6010	8442	6090	1593
21			10	45	226000	324300	972000	357900	15200	21830	6600	2429
21			10	45	227000	324200	1030000	339300	15000	21470	8150	2678
21			0	45	34800	23240	1010000	346700	2980	1978	4160	1429
21			0	55	51000	41930	1150000	357100	3970	3252	4510	1404
30			0	25	47500	4594	685000	299400	1380	134.3	11800	5145
30			0	55	54100	7273	1170000	351100	1890	258.1	8930	2691

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
30			0	50	43200	4355	1100000	350200	1500	154	8330	2662
30			0	35	149000	161200	937000	337100	3550	3859	10400	3729
30			10	40	289000	320200	966000	329100	6470	7171	3760	1283
30			0	50	36100	4314	1060000	356400	1710	207.4	5760	1938
30			10	35	320000	336000	892000	334800	14800	15620	4700	1763
30			30	35	730000	483200	913000	332600	23600	15630	11100	4041
30			20	50	470000	425800	1290000	346100	15900	14390	19500	5221
39			0	15	22300	3567	331000	265200	580	92.42	1170	936
39			0	10	32200	2615	213000	224900	856	69.89	700	739.7
39			0	20	29900	4725	451000	299000	793	125.8	1410	934
39			0	25	29000	2960	538000	320900	635	64.09	1680	999.6
39			0	20	25100	6531	454000	294100	549	141.2	1380	895.6
39			0	10	24000	2601	284000	224700	817	89.12	1180	934
39			0	45	23700	8759	948000	364200	800	296.5	3800	1461
39			0	15	21800	1616	464000	289800	593	43.46	1880	1171
39			0	40	24500	3506	837000	359600	664	94.27	7400	3181

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
6	25	20	0	10	59900	6232	400000	234600	1330	137.6	1630	957
6			0	0	70500	5542	67700	31640	1610	126.4	266	124.4
6			0	15	66100	8133	631000	254200	1490	183.9	2230	898.7
6			0	10	55900	2846	348000	220300	998	50.34	1280	807.7
6			30	15	642000	487700	416000	258100	12100	9187	1440	891.9
6			20	35	443000	427000	772000	345300	13000	12510	3710	1658
6			10	40	422000	357700	1140000	334800	12800	10870	5200	1532
6			0	40	46600	5807	927000	344200	1090	135.4	7290	2708
6			20	30	599000	468700	778000	324200	14000	10950	6700	2792
15			0	0	56000	6839	189000	78260	1700	208.5	756	313.7
15			0	20	68800	10060	542000	289000	2540	375.5	2660	1417
15			0	15	149000	136600	430000	268000	5450	5035	2670	1668
15			0	20	72300	15100	640000	307200	1740	363.3	2250	1079
15			10	15	249000	320100	449000	250900	6050	7781	1500	841
15			0	10	69600	31370	435000	247500	3530	1605	2050	1167
15			20	30	449000	425700	835000	310000	23000	21840	3810	1413

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size	Links	MIMIC	GA Not	MIMIC	+/-	GA	+/-	MIMIC	+/-	GA	+/-
		Active	Not Conv	Conv	Calls		Calls		time		time	
	(inputs)	%	%	%					ms	ms	ms	ms
15			10	10	244000	321100	367000	230800	7310	9627	1680	1054
15			0	25	234000	307600	661000	303500	6560	8641	2950	1355
24			0	5	29800	3328	306000	193400	722	81.08	1530	963.2
24			0	15	98800	95530	453000	269500	2540	2464	5560	3311
24			10	45	233000	323000	1220000	316600	5950	8247	8020	2076
24			0	25	37700	3063	676000	329300	759	61.18	6630	3229
24			20	20	428000	431200	646000	281700	14500	14550	6720	2927
24			20	15	471000	425700	565000	294100	24700	22320	7810	4065
24			10	35	227000	324200	925000	340200	10800	15440	8100	2978
24			0	15	43200	30320	459000	266200	4850	3389	2750	1596
24			20	35	581000	446100	825000	330500	25700	19700	4720	1889
33			0	0	33800	2903	38500	23340	1010	86.81	178	107.7
33			0	0	40100	4967	56500	43150	1190	148.2	233	177.6
33			0	0	35900	3755	148000	114500	1070	112.7	575	445.2
33			0	0	40800	4328	141000	159400	962	101.5	548	619.1
33			0	0	195000	264400	111000	117600	5700	7724	432	457.1

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
33			0	10	28100	3064	301000	241100	1870	205.2	1540	1238
33			0	15	234000	305300	483000	279000	11800	15410	2380	1375
33			0	5	31200	6558	276000	213200	4150	866.5	1400	1078
33			10	20	266000	323600	529000	287900	31700	38590	2540	1381
42			0	0	36900	3134	170000	144200	1100	94.68	736	624.3
42			10	5	244000	321100	205000	193100	9240	12210	794	750.3
42			0	5	47300	3873	148000	166100	1720	143.4	568	638.7
42			0	0	49300	5453	91900	45050	1180	130.7	361	176.7
42			10	0	235000	322600	78600	42570	5790	7935	302	163.5
42			0	0	66100	38540	91200	67340	3220	1891	474	349.7
42			0	5	35800	3965	265000	216800	1830	205.8	1300	1066
42			10	0	227000	324000	138000	92710	6950	9915	697	469
42			0	20	35400	4178	542000	289400	1010	119.7	2600	1388
1	50	5	0	0	59500	5229	12600	1093	12000	1052	153	13.24
1			0	0	97300	14980	16700	3947	7650	1176	179	42.21
1			0	0	69700	7002	15700	2231	5430	544.1	168	23.93

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
1			0	0	83100	16690	15800	1424	5990	1196	164	14.81
1			0	0	73600	23900	16300	1562	5300	1710	164	15.68
1			0	0	69100	17310	41300	43620	6590	1648	440	465.3
1			0	0	71100	16970	16200	4507	6710	1598	195	54.15
1			0	0	48300	7078	24300	8239	4040	587.8	265	89.83
1			0	0	69700	39200	24500	9157	5780	3235	259	96.88
10			0	45	455000	76720	1240000	321300	23700	3986	10200	2645
10			20	50	881000	311300	1220000	330900	48000	16950	9130	2482
10			20	50	777000	341400	1280000	308100	42200	18540	23400	5626
10			30	40	1180000	374000	1010000	323600	42700	13560	22100	7089
10			20	60	698000	360900	1410000	289900	25500	13130	21300	4394
10			50	60	1190000	444700	1410000	292400	87700	32680	33800	7003
10			10	70	602000	287400	1580000	274500	44100	21040	33500	5832
10			40	40	1080000	416400	1160000	321400	58300	22460	26300	7292
10			0	55	395000	113600	1290000	304800	21200	6079	12000	2826
19			20	50	837000	348200	1240000	298400	67400	28040	33300	7987

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
19			40	70	1070000	434700	1590000	244800	88500	36100	13300	2048
19			60	95	1520000	378300	1940000	103600	310000	77200	35000	1873
19			70	65	1740000	252000	1540000	280700	268000	38770	29800	5445
19			50	75	1160000	472500	1600000	263800	196000	79620	35800	5905
19			40	85	958000	469100	1830000	173200	284000	138800	20500	1946
19			70	95	1450000	465200	1960000	73620	349000	112400	21100	794.4
19			80	75	1630000	407000	1650000	242600	303000	75660	18200	2683
19			50	75	1170000	459400	1680000	235400	686000	269900	95500	13380
28			20	55	1080000	330300	1240000	325300	120000	36570	14000	3687
28			40	60	1400000	325400	1280000	334000	88100	20510	13000	3391
28			30	80	1260000	291400	1670000	253100	144000	33130	42800	6468
28			70	65	1730000	244600	1430000	300200	142000	20010	41000	8598
28			50	60	1440000	326200	1650000	208200	128000	29120	41700	5254
28			40	50	1320000	310700	1240000	314200	219000	51550	51600	13050
28			70	70	1700000	273800	1500000	297000	245000	39540	47300	9402
28			20	55	1210000	330900	1280000	325500	163000	44470	51600	13160

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
28			60	65	1500000	350900	1590000	248900	412000	96550	20800	3261
37			40	55	1180000	406200	1420000	290000	197000	68130	71400	14600
37			50	80	1290000	390700	1690000	238700	83600	25230	25200	3572
37			60	85	1430000	388000	1730000	239300	206000	55890	24900	3428
37			60	90	1560000	351700	1830000	195300	182000	41100	27300	2921
37			60	70	1390000	421600	1560000	257500	175000	53040	22200	3664
37			70	75	1590000	351200	1580000	282000	333000	73660	32800	5871
37			60	100	1500000	372800	2000000	0	331000	82240	40100	0
37			90	85	1820000	297100	1780000	214300	301000	49050	36700	4415
37			80	80	1710000	325500	1640000	274900	112000	21370	70000	11750
4	50	10	0	55	577000	203200	1360000	297100	41500	14640	31300	6858
4			10	75	777000	245200	1630000	247200	60700	19170	32900	4976
4			10	70	647000	255500	1640000	222200	50400	19920	31900	4319
4			0	90	496000	108300	1810000	214500	22800	4974	36700	4344
4			20	60	742000	348100	1540000	244800	33900	15890	29800	4729
4			30	75	1070000	349900	1680000	226700	98100	32190	48000	6463

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size	Links	MIMIC	GA Not	MIMIC	+/-	GA	+/-	MIMIC	+/-	GA	+/-
		Active	Not Conv	Conv	Calls		Calls		time		time	
	(inputs)	%	%	%					ms	ms	ms	ms
4			20	85	901000	326300	1770000	215100	83300	30180	48800	5916
4			20	80	834000	345100	1780000	202700	56600	23390	50400	5755
4			30	75	970000	428600	1670000	254000	65600	28990	101000	15290
13			0	20	360000	43400	959000	277500	28100	3385	9960	2883
13			10	55	720000	243100	1410000	276500	56400	19010	13100	2572
13			0	80	501000	65000	1690000	236200	39200	5092	15200	2123
13			10	70	721000	270200	1620000	248500	46000	17210	14800	2282
13			20	80	701000	358200	1620000	284000	44300	22610	14500	2536
13			10	70	518000	275400	1620000	245400	48100	25560	20400	3095
13			20	95	693000	363600	1910000	147200	64000	33580	23400	1802
13			40	70	1040000	433300	1480000	302400	82200	34240	18500	3780
13			20	65	996000	372200	1580000	266700	77700	29010	19100	3229
22			0	0	180000	29210	61200	21400	58600	9529	714	249.6
22			0	0	291000	39180	156000	89050	45700	6155	1560	886.8
22			0	5	254000	49240	431000	236300	38300	7440	4020	2201
22			0	10	265000	52360	327000	222500	48900	9645	3180	2161

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
22			0	10	197000	45110	276000	217500	42700	9791	2530	2000
22			10	5	402000	294500	198000	162600	189000	138900	7930	6528
22			10	15	403000	298900	564000	258100	188000	139800	13000	5960
22			10	10	450000	314300	424000	241400	116000	80920	11100	6340
22			10	25	488000	355900	736000	290500	519000	379100	132000	51930
31			0	55	335000	37080	1260000	317700	24100	2668	37800	9549
31			20	55	720000	355400	1290000	314200	56200	27740	21200	5161
31			20	65	695000	360400	1430000	298000	77200	40090	14700	3061
31			10	65	555000	269900	1450000	298700	45800	22200	15600	3221
31			0	60	454000	144100	1470000	262500	20800	6617	15000	2676
31			20	60	727000	352400	1490000	286700	121000	58470	69300	13310
31			20	85	658000	369800	1800000	188300	116000	65270	70200	7367
31			10	85	617000	337700	1820000	201800	42100	23030	70700	7853
31			20	80	615000	382000	1720000	241800	41700	25870	23900	3368
40			20	55	828000	363400	1430000	264800	40100	17580	74900	13910
40			20	60	849000	326600	1360000	308400	39600	15190	13800	3140

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
40			50	65	1210000	433200	1490000	276000	55700	19890	13200	2444
40			60	55	1380000	419500	1270000	315000	49600	15090	11700	2907
40			60	60	1380000	420400	1530000	238100	49000	14900	13500	2090
40			40	60	1030000	437900	1500000	273200	70900	30230	54000	9811
40			20	60	682000	363000	1590000	218000	47200	25090	54000	7395
40			30	70	1090000	410800	1660000	251300	58200	21960	52300	7915
40			80	90	1690000	350800	1880000	140100	89000	18520	22100	1649
7	50	20	0	0	214000	31450	122000	31330	16600	2433	2010	513.9
7			0	0	283000	34370	160000	54620	21200	2571	6980	2380
7			0	5	227000	25170	327000	203800	16900	1872	4540	2827
7			0	5	318000	54970	296000	166800	18700	3228	4240	2389
7			10	10	412000	290900	368000	214900	23600	16610	4970	2905
7			0	10	231000	28730	363000	232800	20900	2592	7010	4501
7			10	5	370000	300600	325000	195200	31900	25970	5990	3593
7			20	0	576000	394700	299000	121600	44800	30700	5720	2327
7			10	15	360000	300700	677000	270900	27900	23250	128000	51210

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
16			0	60	1230000	172600	1430000	279000	189000	26610	23200	4509
16			50	45	1850000	93570	1210000	296600	130000	6550	17500	4297
16			0	55	1250000	155500	1510000	240000	87600	10910	21000	3331
16			20	80	1510000	187600	1700000	228700	63600	7887	24500	3295
16			0	65	911000	82300	1440000	293500	38300	3458	20000	4068
16			10	45	1500000	175700	1180000	299300	133000	15630	23200	5854
16			0	90	1050000	125100	1860000	164400	94100	11270	34700	3072
16			10	45	1380000	175400	1310000	271200	87500	11120	25800	5331
16			0	75	853000	212600	1620000	254400	53000	13200	30500	4785
25			0	20	313000	38910	875000	265600	12800	1585	13400	4051
25			0	25	464000	50510	823000	305100	22100	2407	11400	4219
25			10	35	581000	264200	970000	324900	22100	10050	12700	4248
25			0	20	343000	50820	602000	275800	10500	1556	8210	3762
25			0	40	454000	113400	1120000	315300	13900	3462	14600	4110
25			30	45	999000	423500	1210000	309400	89300	37820	50300	12870
25			0	50	263000	35610	1230000	307400	34600	4680	49900	12530

Table 37: Detailed Static Comparison Results Per Problem (+/-20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
25			20	15	615000	380800	560000	256500	63000	38930	46700	21380
25			30	55	892000	402900	1270000	316200	246000	111000	323000	80480
34			0	75	1260000	100500	1720000	202400	146000	11650	26400	3115
34			70	90	1870000	123500	1830000	199100	97400	6393	25000	2726
34			10	80	1290000	166200	1770000	214300	62000	7982	23100	2806
34			60	75	1840000	145200	1770000	171400	70300	5520	24400	2364
34			30	65	1540000	210300	1520000	302700	58200	7952	19900	3964
34			0	80	1800000	76770	1720000	213400	301000	12870	88800	11000
34			30	95	1280000	276800	1960000	74460	262000	56530	86800	3307
34			50	65	1700000	189400	1380000	320900	267000	29600	87000	20270
34			30	80	1250000	321600	1800000	200800	70000	18000	636000	70860
43			0	35	678000	172400	1140000	299400	50700	12900	15400	4030
43			10	30	959000	210400	855000	296000	78400	17200	10400	3602
43			10	55	812000	224600	1400000	275600	66000	18290	16200	3196
43			10	35	857000	232100	1140000	280900	41200	11150	14000	3437
43			0	35	542000	73160	1120000	278600	26100	3526	13100	3238

Table 37: Detailed Static Comparison Results Per Problem (+/- 20% from Best Method Settings)

ProbID	Size (inputs)	Links Active %	MIMIC Not Conv %	GA Not Conv %	MIMIC Calls	+/-	GA Calls	+/-	MIMIC time ms	+/- ms	GA time ms	+/- ms
43			20	65	1050000	329300	1540000	270300	101000	31820	64800	11400
43			0	50	517000	78190	1400000	294600	50500	7648	60500	12720
43			0	50	558000	61350	1280000	286200	40000	4391	63600	14240
43			0	65	455000	101900	1510000	259900	32300	7217	267000	45930

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	0	25	30	5	2
Problem	-1 -1 -1 14 5 2 7 -1 -1 10 15 3 11 -1 10 3 15 7 23 9 -1 -1 -1 -1 9 13 0 19 -1 -1 10 -1 11 24 -1 15 -1 -1 3 -1 -1 -1 -1 2 10 12 11 21 -1 -1 -1 -1 10 15 -1				
	1	50	122	5	13
Problem	22 -1 3 13 -1 41 37 -1 41 21 -1 40 20 31 -1 6 -1 20 -1 5 32 36 -1 -1 15 32 21 48 47 10 3 49 35 22 -1 -1 -1 47 13 5 17 36 23 -1 -1 8 49 -1 -1 48 35 -1 2 8 30 43 26 19 -1 32 22 33 37 12 11 -1 -1 36 -1 28 11 -1 -1 42 11 35 33 -1 22 5 34 25 -1 -1 16 -1 6 48 40 35 20 45 12 49 42 -1 35 -1 1 18 16 0 20 -1 45 -1 1 44 38 -1 0 36 48 35 42 14 -1 8 14 45 -1 -1 30 37 1 41 10 -1 -1 12 20 38 9 39 13				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	30 -1 -1 5 42 -1 9 -1 9 44 -1 -1 19 26 40 -1 -1 22 18 40 23 -1 -1 10 -1 19 -1 1 29 3 42 27 2 38 47 9 -1				
	3	25	60	10	6
Problem	24 15 -1 -1 9 3 4 6 -1 -1 -1 -1 -1 15 0 19 16 5 10 -1 -1 22 16 15 7 -1 11 16 -1 17 21 19 8 -1 8 23 4 19 -1 3 2 -1 -1 17 0 18				
	-1 13 3 12 22 4 1 5 19 23 -1 -1 2 -1 9 11 13 5 -1 0 10 2 13 -1 0 19 1 17 -1 11 13 -1 18 19 6 7 -1 23 -1				
	4	50	245	10	40
Problem	-1 36 5 18 8 -1 45 -1 48 33 16 9 -1 1 11 38 25 3 -1 20 27 24 29 37 25 13 15 14 22 46 21 33 -1 12 29 37 24 16 -1 -1 28 15 4 -1				
	4 18 26 11 36 41 24 2 17 39 13 12 35 46 25 10 -1 -1 -1 19 1 7 -1 37 -1 0 30 21 42 5 6 1 10 38 4 43 37 39 -1 3 2 25 16 6 27 40				
	28 10 47 21 -1 6 14 27 15 31 26 32 5 23 33 28 22 0 -1 36 34 31 -1 19 37 38 17 16 11 -1 23 9 -1 11 -1 44 10 25 31 18 22 8 14 -1				
	10 -1 44 31 34 40 2 14 46 3 6 32 29 22 0 25 11 39 26 35 37 -1 23 4 28 12 33 -1 8 40 -1 38 37 -1 26 35 11 4 9 42 -1 -1 38 34 44				
	25 24 26 12 30 18 -1 -1 17 40 34 5 38 -1 47 40 25 33 49 23 -1 47 41 14 -1 33 3 9 24 26 19 4 17 25 -1 -1 -1 32 48 28 13 2 -1 8				
	22 41 32 2 26 44 16 47 25 31 11 33 0 15 12 -1 -1 17 21 27 39 31 20 29 23 -1 45 2 28 39 35 23 10 31 -1 19 29 -1 38 -1 14 29 -1				
	48 2 -1 3 21 48 40 35 39 -1 16 -1 15 25 -1 10 32 29 27 3 39 36 42 44 20 34 6 14 -1				
	6	25	120	20	21
Problem	5 10 2 9 -1 22 15 0 9 24 -1 1 3 6 14 22 20 9 -1 -1 11 -1 14 12 -1 2 0 21 10 20 9 8 7 4 -1 23 9 -1 4 12 3 19 5 18 2 -1 5 -1 4 6				
	8 -1 20 17 6 9 4 -1 1 17 9 24 5 15 -1 20 11 3 22 -1 11 20 15 21 -1 2 16 4 24 -1 10 6 22 3 5 -1 7 4 10 19 20 18 22 2 -1 10 19 2				
	1 6 7 17 13 23 24 21 12 -1 22 7 10 9 11 20 2 0 1 -1 9 14 1 13 21 4 15 -1 5 4 8 14 -1 1 -1 11 14 16 5 -1 3 13 18 10 9 4 -1				
	7	50	490	20	88
Problem	23 7 46 13 31 5 17 14 11 36 26 -1 49 16 37 11 14 -1 13 41 24 21 35 40 29 -1 25 29 11 0 -1 26 0 12 14 36 -1 -1 13 44 19 12 4 18				
	42 47 -1 -1 12 19 4 30 48 28 44 26 21 24 5 49 46 3 0 17 41 27 6 35 -1 29 4 14 48 37 49 38 36 23 8 11 25 19 7 0 15 10 33 41 2				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	24 46 12 43 34 40 45 20 27 6 28 16 32 26 -1 15 40 26 8 48 6 12 22 9 16 3 5 27 4 17 14 2 0 42 25 36 33 19 47 1 32 7 38 31 37 11 39 21 -1 25 -1 28 48 39 9 6 14 45 43 16 41 -1 41 2 25 28 -1 48 23 21 -1 33 39 7 32 38 -1 5 23 33 21 20 25 11 28 39 45 2 -1 47 41 8 48 -1 4 23 32 39 17 13 19 44 14 11 22 48 47 49 15 8 7 9 30 20 12 5 26 41 42 46 38 29 43 33 -1 33 -1 31 26 17 45 47 32 28 46 49 2 13 30 42 12 8 24 38 -1 -1 23 46 31 43 9 5 28 6 39 18 38 -1 41 20 2 22 48 7 33 11 42 -1 28 34 40 -1 0 29 41 48 -1 27 34 38 44 24 10 12 2 25 -1 3 41 0 13 1 46 40 38 32 30 29 43 -1 0 14 20 40 19 5 47 18 49 46 38 23 27 17 42 3 37 35 31 21 -1 7 31 37 11 14 9 -1 20 35 15 33 17 45 21 12 7 38 8 34 16 49 42 5 6 -1 5 32 39 27 16 29 6 45 8 26 49 30 41 40 20 15 12 48 34 7 24 -1 9 37 1 2 22 -1 12 37 14 1 21 5 36 13 9 7 34 49 35 -1 13 47 23 2 15 33 48 29 36 32 9 22 -1 4 31 -1 26 -1 39 13 3 1 8 -1 41 18 37 28 30 32 22 9 14 19 46 40 34 2 45 15 13 25 10 24 44 31 48 7 26 29 3 39 49 6 17 20 5 4 1 8 0 36 23 47 35 12 11 33 -1 49 48 -1 7 10 3 28 36 0 17 48 9 47 46 16 26 -1 -1 38 12 14 25 39 -1 16 6 19 25 14 35 23 22 47 17 -1 10 9 12 8 40 31 33 14 2 20 37 30 6 43 34 -1 38 11 49 2 32 -1 49 42 16 35 7 -1 45 19 42 8 24 46 2 25 27 -1 34 45 8 1 0 38 41 30 21 29 10 -1 46 2 34 41 14 48 7 5 -1				
Problem	9	25	30	5	1
	21 -1 13 20 -1 6 15 18 -1 -1 -1 -1 -1 2 -1 10 -1 3 24 11 -1 18 -1 9 24 16 2 -1 -1 -1 -1 12 18 14 -1 -1 -1 -1 10 20 16 -1 10 14 16 3 -1 23 6 -1 14 23 -1 -1 -1				
Problem	10	50	122	5	13
	39 -1 -1 45 19 4 15 3 49 32 16 40 17 46 -1 1 24 -1 -1 19 25 14 28 18 38 27 33 31 -1 24 46 5 43 23 -1 9 20 17 -1 -1 16 27 11 26 43 22 46 17 -1 39 -1 3 42 -1 15 13 -1 29 10 27 -1 46 40 23 5 34 -1 44 30 19 14 21 1 -1 13 7 47 -1 13 16 38 2 -1 -1 -1 -1 -1 48 26 -1 -1 45 -1 31 16 40 -1 28 -1 -1 30 -1 36 34 41 -1 36 -1 29 16 5 9 43 21 25 3 -1 39 22 29 -1 42 -1 25 1 32 44 5 17 -1 9 46 -1 37				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	-1 22 32 -1 37 45 6 -1 10 37 26 5 44 16 -1 32 46 26 48 -1 -1 -1 7 47 -1 -1 6 3 -1 41 30 21 -1 1 -1 -1 25 -1				
Problem	12	25	60	10	7
	-1 12 -1 17 -1 5 20 14 17 16 -1 10 17 -1 8 22 -1 10 18 -1 16 9 23 6 -1 12 1 17 18 2 21 -1 4 -1 9 16 -1 20 -1 -1 6 -1 24 -1 14 0 8 24 -1 -1 4 0 3 20 -1 16 23 13 17 -1 21 8 23 17 13 3 16 9 7 0 2 -1 23 -1 5 -1 -1 20 5 10 21 6 8 -1 -1				
Problem	13	50	245	10	36
	46 19 22 -1 -1 7 -1 14 36 1 17 41 22 30 24 8 19 34 37 -1 20 16 32 7 10 37 35 2 -1 0 4 41 22 -1 17 29 19 34 44 33 37 35 13 12 45 9 5 -1 -1 -1 16 2 29 34 44 23 -1 3 19 6 27 -1 38 -1 24 32 -1 -1 42 46 28 34 13 15 37 4 30 33 45 36 11 2 32 0 -1 31 47 22 32 44 37 17 8 16 34 24 -1 4 26 2 31 38 6 20 45 37 49 44 25 33 39 -1 15 38 27 20 5 22 16 28 -1 46 11 15 24 -1 35 17 -1 2 9 49 -1 10 8 14 26 -1 42 48 13 -1 -1 10 -1 -1 12 -1 34 33 4 18 36 39 47 11 -1 23 0 14 47 -1 30 16 11 33 47 -1 46 -1 46 38 25 -1 31 24 -1 -1 28 47 14 4 -1 37 34 47 32 5 31 49 2 3 38 40 48 -1 -1 30 12 48 -1 6 17 20 25 26 16 41 15 -1 15 23 3 49 37 40 44 12 -1 34 14 19 -1 44 14 20 32 7 23 22 47 31 -1 0 -1 13 32 8 48 9 45 -1 -1 18 14 38 26 1 37 3 11 12 4 -1 43 23 10 42 24 1 35 30 49 18 -1 42 3 4 9 37 -1 45 6 34 39 27 4 14 30 28 33 46 41 25 35 49 42 15 18 9 -1 41 30 14 -1				
Problem	15	25	120	20	25
	12 5 16 19 -1 10 21 6 7 18 -1 24 9 6 0 17 14 8 -1 7 12 16 17 24 -1 7 24 19 17 6 15 9 2 22 -1 10 4 16 3 -1 21 11 -1 0 22 10 6 16 8 20 3 5 2 14 -1 2 9 24 21 18 6 -1 23 24 15 -1 6 8 -1 -1 -1 7 -1 21 1 20 18 19 -1 6 20 16 1 2 -1 12 19 10 -1 -1 3 12 9 23 6 22 14 1 7 19 15 16 -1 6 9 5 21 24 14 1 -1 11 18 5 12 -1 9 6 22 0 20 10 23 11 4 13 -1 11 13 7 2 8 -1 7 -1 14 6 5 21 15 12 19 18 7 -1				
Problem	16	50	490	20	95
	28 20 14 -1 4 29 34 42 5 28 30 12 2 24 26 9 23 32 33 27 48 49 40 -1 9 23 0 3 41 47 35 19 43 28 14 21 33 36 4 18 32 6 11 20 29 34 38 1 7 39 48 10 45 5 15 25 -1 -1 -1 9 34 27 49 -1 14 -1 36 18 19 39 8 48 9 47 45 -1 11 48 17 41 13 28 33 26 36 20 18 44 43				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	27 -1 19 28 48 38 39 16 21 24 8 2 -1 37 19 8 41 0 23 32 16 35 12 33 30 14 2 39 47 15 49 9 -1 27 13 7 -1 47 6 4 35 25 -1 -1 48 18 40 23 17 47 9 15 7 35 26 20 2 25 33 42 16 41 27 4 28 43 22 34 36 45 0 12 29 37 5 39 8 1 32 30 38 6 10 11 19 -1 20 -1 40 43 8 18 36 47 14 12 29 25 24 1 6 26 4 28 46 19 27 22 5 32 45 21 42 34 31 20 23 0 3 30 -1 14 1 45 22 19 41 32 28 31 40 48 30 11 18 -1 12 16 38 6 -1 39 9 33 46 26 -1 16 25 19 36 44 0 47 40 33 30 13 29 35 3 11 28 8 5 31 37 14 24 26 34 23 43 18 -1 36 0 32 47 45 29 15 28 6 12 39 49 -1 16 46 12 24 17 41 6 40 39 -1 37 4 39 33 32 13 25 49 -1 25 45 23 28 22 -1 11 -1 32 10 25 48 24 35 22 -1 37 20 1 3 46 23 48 8 -1 -1 20 16 0 5 -1 9 41 -1 21 10 34 -1 49 36 10 11 14 41 5 27 21 25 12 29 20 23 37 33 48 1 43 16 26 -1 32 14 23 36 11 -1 46 40 21 35 18 29 24 6 -1 46 38 47 21 43 27 2 7 37 33 30 32 29 24 49 5 20 36 -1 27 41 1 4 18 29 7 45 -1 16 32 8 33 22 14 4 -1 40 47 13 29 19 33 41 46 2 -1 44 30 23 9 29 15 37 42 25 7 21 43 27 -1 34 -1 0 31 42 48 16 26 30 17 47 34 40 19 45 43 15 9 14 39 -1 38 0 40 21 11 25 18 22 37 -1 2 41 19 45 35 14 12 3 18 23 -1 37 38 33 28 49 40 5 45 35 24 26 27 3 10 47 23 39 1 -1 15 10 27 4 14 38 18 31 6 2 30 0 16 40 28 24 32 37 -1 37 40 20 17 25 14 -1 -1 47 18 4 29 42 30 43 21 37 6 26 24 -1 39 25 13 6 17 23 16 -1				
	18	25	30	5	1
Problem	-1 11 5 -1 18 9 -1 10 6 -1 -1 16 11 -1 -1 2 -1 -1 -1 -1 15 -1 16 19 7 -1 -1 -1 11 22 -1 -1 -1 15 -1 -1 16 3 6 -1 19 18 10 3 -1 -1 16 24 0 -1 21 9 20 17 -1				
	19	50	122	5	11
Problem	20 6 2 29 16 37 41 -1 2 11 17 22 -1 43 24 -1 -1 19 -1 39 21 10 11 0 -1 48 -1 45 27 6 -1 16 36 40 26 23 5 17 -1 -1 -1 13 31 21 30 35 -1 -1 4 5 0 31 42 -1 29 16 1 19 11 8 42 27 5 -1 -1 -1 36 0 3 -1 23 11 26 6 -1 43 -1 31 -1 -1 38 -1 -1 41 33 1 23 3 14 -1 40 -1 27 -1 33 19 35 20 43 -1 16 10 44 42 12 -1 24 -1 48 45 -1 19 -1 48 33 26 45 15 44 39 -1 -1 -1 -1 -1 1 29 31 28 -1 21 25 -1 26 14				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	-1 37 49 -1 40 14 19 -1 -1 31 18 41 -1 45 -1 30 39 24 -1 38 3 42 9 39 8 16 17 6 24 13 -1 41 -1 -1 2 29 -1				
	21	25	60	10	5
Problem	11 -1 17 20 21 0 -1 -1 23 0 -1 2 -1 -1 24 -1 8 22 13 -1 -1 19 2 7 6 22 16 14 -1 7 21 -1 -1 -1 6 17 20 18 23 7 21 -1 19 2 -1 5 13 21 18 23 0 16 24 6 19 -1 -1 19 13 2 24 12 5 7 -1 -1 1 11 8 20 -1 -1 18 17 -1 5 -1 18 24 12 -1 6 1 10 -1				
	22	50	245	10	33
Problem	28 23 46 47 37 48 -1 42 40 -1 8 37 28 9 48 39 47 33 42 36 45 -1 29 -1 11 38 3 -1 31 49 13 42 18 25 44 -1 43 28 0 39 -1 -1 26 39 43 29 7 25 46 13 34 12 38 6 0 35 -1 -1 13 30 -1 -1 17 -1 30 45 19 6 -1 42 -1 46 32 14 44 2 25 29 20 -1 8 44 9 7 42 -1 39 -1 8 29 14 16 3 20 35 41 25 -1 25 35 16 -1 19 35 1 -1 18 11 1 -1 49 15 -1 40 19 12 37 2 8 47 21 26 20 10 43 1 -1 28 2 25 19 44 18 42 6 35 46 5 -1 40 46 29 -1 46 15 -1 1 -1 14 48 36 6 27 2 9 1 49 -1 16 27 19 -1 47 4 24 -1 5 39 9 47 36 42 26 32 22 17 34 13 0 49 19 4 43 -1 42 2 7 -1 29 42 4 7 44 -1 26 43 1 27 39 32 42 5 24 11 35 25 20 19 38 31 33 9 49 46 -1 47 19 6 10 23 48 2 11 9 31 24 45 43 44 15 4 16 12 40 0 26 17 8 30 49 -1 17 29 8 28 -1 8 -1 48 14 36 19 30 -1 -1 19 8 28 -1 26 8 23 -1 31 -1 28 27 -1 -1 14 20 46 -1 22 29 7 49 -1 49 17 -1 1 13 4 28 49 47 16 -1 13 11 34 30 19 -1				
	24	25	120	20	16
Problem	23 13 -1 5 21 -1 13 9 17 0 23 11 6 18 4 15 5 19 1 12 -1 21 -1 -1 -1 8 7 4 21 12 5 3 16 20 24 11 15 -1 13 17 18 21 1 -1 3 -1 15 14 21 5 24 20 10 2 22 16 8 23 6 13 1 11 17 -1 12 -1 24 23 21 5 13 14 -1 21 23 20 11 3 -1 14 -1 19 0 12 2 15 6 1 23 10 -1 3 17 -1 17 4 10 1 8 23 18 6 5 -1 -1 19 20 15 10 3 -1 -1 -1 22 5 24 13 9 6 11 10 0 20 15 14 4 12 7 3 23 2 -1 -1 1 15 24 11 -1 9 15 19 11 0 6 -1				
	25	50	490	20	98
Problem	21 3 34 13 -1 38 21 12 23 6 0 16 33 41 36 17 -1 16 18 32 13 4 -1 20 10 46 -1 38 24 10 31 45 40 1 30 32 43 12 15 19 27 -1 21 8				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	49 -1 23 -1 49 -1 49 12 14 34 44 -1 19 42 43 -1 0 40 49 4 48 26 -1 45 9 10 31 -1 49 47 46 48 23 26 40 20 7 -1 -1 8 11 21 43 38 -1 32 14 9 24 43 39 35 7 41 -1 19 43 7 4 40 39 27 6 12 17 9 42 33 46 15 37 48 28 24 38 -1 43 46 10 20 22 13 48 14 40 36 1 44 33 18 0 19 25 32 23 38 4 29 35 12 2 3 15 39 45 26 -1 20 5 23 13 29 43 41 6 -1 45 -1 48 40 22 46 33 5 18 39 26 37 43 17 19 47 6 3 24 34 38 4 41 14 29 8 16 44 31 10 36 30 32 0 35 -1 39 -1 31 41 40 45 20 37 39 3 36 24 13 25 47 48 21 -1 33 39 47 0 19 9 31 30 18 24 6 16 26 40 28 21 7 5 34 46 45 -1 49 9 12 42 47 3 20 37 8 30 2 7 19 4 5 33 13 23 40 14 44 39 32 6 10 46 11 28 35 36 -1 24 20 38 49 34 15 -1 -1 -1 29 38 7 44 14 4 -1 34 44 45 15 23 36 -1 -1 23 12 44 26 3 25 27 2 8 0 34 33 48 28 18 11 1 49 32 47 10 45 -1 22 6 21 17 40 5 -1 43 16 -1 14 24 35 18 9 0 27 4 25 37 7 39 44 28 19 -1 31 -1 47 27 23 28 22 19 9 31 42 11 17 45 0 32 40 12 2 5 18 7 46 20 13 48 43 29 38 6 25 16 -1 28 15 16 32 21 14 31 33 43 22 41 24 2 0 39 -1 46 14 35 44 20 33 26 19 6 23 -1 5 41 46 30 16 0 48 17 24 15 13 12 27 25 14 45 47 10 20 8 32 44 43 1 4 31 26 18 36 7 21 -1 29 8 46 30 35 11 41 34 -1 11 8 9 0 10 37 -1 28 5 45 -1 27 -1 33 14 49 41 47 29 0 27 4 37 42 16 26 3 38 6 32 9 17 46 21 20 39 23 -1 27 22 8 20 16 32 26 12 6 7 19 25 0 41 40 31 30 21 23 10 -1 42 3 29 7 31 13 35 9 17 43 1 45 36 37 16 4 0 38 20 24 -1 41 2 13 25 1 35 46 33 17 -1 -1 24 17 21 25 11 43 10 -1				
	27	25	30	5	0
Problem	21 19 -1 5 23 16 -1 14 -1 -1 -1 -1 -1 13 2 9 14 10 1 -1 17 22 -1 10 16 -1 14 18 20 -1 -1 5 22 -1 21 8 -1 -1 2 5 -1 -1 23 9 -1 11 -1 -1 12 -1 -1 -1 -1 13 -1				
	28	50	122	5	11
Problem	12 13 33 -1 -1 -1 31 5 32 -1 40 -1 14 -1 42 -1 32 8 0 33 17 48 43 19 13 -1 49 36 -1 15 17 43 12 45 35 30 -1 30 15 26 16 17 -1 46 15 -1 13 32 -1 4 40 10 -1 15 13 49 -1 -1 6 28 42 46 -1 4 -1 29 -1 40 39 24 36 -1 -1 18 47 -1 34 49 -1 26 36 14 37 9 -1 28 11 2 20 29 31 9 22 0 21 33 7 1 41 14 -1 -1 -1 39 -1 11 -1 21 -1 0 47 35 -1 44 43 38 26 28 41 49 0 19 46 25 9 -1 5 -1 16 -1 46 49 32 -1 -1				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	-1 -1 31 -1 -1 47 46 44 9 41 -1 49 -1 -1 5 11 4 9 49 -1 -1 37 26 35 -1 40 -1 31 -1 -1 37 15 44 14 45 31 -1				
	30	25	60	10	5
Problem	21 14 19 24 11 7 9 -1 10 -1 15 12 -1 -1 18 23 11 22 -1 -1 22 -1 8 14 15 3 11 -1 20 2 24 -1 2 -1 -1 -1 20 19 0 -1 1 -1 18 12 4 13				
	10 8 -1 11 20 13 1 14 22 17 24 -1 10 22 -1 5 14 22 15 20 16 -1 9 19 15 10 20 -1 4 -1 -1 -1 5 -1 -1 15 3 12 -1				
	31	50	245	10	35
Problem	45 5 32 15 46 -1 39 32 22 27 -1 21 8 19 47 37 44 -1 25 40 36 -1 3 38 -1 2 26 -1 37 42 21 -1 -1 4 21 6 14 0 -1 -1 41 -1 18 7 29				
	22 35 17 26 33 23 12 40 10 41 -1 46 11 -1 3 5 11 4 -1 29 37 2 11 17 41 -1 -1 13 34 14 0 20 26 41 17 39 42 9 25 1 29 23 -1 33 -1				
	16 1 8 44 31 29 43 19 33 -1 20 28 18 15 17 23 -1 30 -1 25 49 -1 -1 45 2 29 -1 31 18 -1 -1 -1 24 -1 8 34 24 42 47 38 17 6 30 1 3				
	46 35 0 40 45 -1 8 -1 42 4 37 24 49 8 47 12 22 13 18 5 38 46 25 11 32 26 33 0 10 23 1 48 39 -1 43 -1 3 31 8 0 6 -1 39 43 8 -1				
	37 1 15 17 49 -1 2 -1 16 11 19 17 31 14 -1 47 36 28 30 41 35 12 -1 44 -1 12 34 31 43 5 11 16 30 15 2 44 8 42 9 -1 49 23 44 16				
	36 9 46 39 25 -1 31 -1 33 -1 4 34 20 21 22 35 3 32 0 28 9 37 1 16 -1 43 37 4 31 48 29 33 14 49 10 5 15 26 7 32 47 24 19 -1 11				
	7 34 3 44 4 21 36 10 -1 37 44 9 39 2 42 29 43 -1 -1 -1 28 41 43 45 -1				
	33	25	120	20	19
Problem	4 9 12 22 21 -1 22 -1 6 15 8 21 -1 21 13 10 24 15 14 11 22 20 -1 -1 0 -1 11 10 5 23 21 8 -1 11 10 15 22 9 12 14 24 13 23 2 20				
	17 -1 22 4 6 16 0 13 19 14 12 7 24 20 1 3 17 11 -1 2 22 12 18 21 15 3 7 10 16 8 -1 14 11 4 13 23 19 8 3 -1 17 19 18 -1 17 2 19				
	-1 -1 4 16 5 -1 3 9 16 8 7 10 -1 20 11 12 14 18 3 10 2 -1 7 19 22 16 12 14 24 -1 2 13 14 7 11 -1 15 -1 15 13 -1 -1 12 5 4 -1 -1				
	10 4 13 20 11 -1				
	34	50	490	20	88
Problem	16 4 -1 7 12 41 -1 23 19 45 42 14 30 37 -1 48 29 11 1 0 2 -1 9 7 42 20 21 3 25 -1 26 49 0 8 14 16 31 33 11 35 39 4 43 1 48 27				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	28 13 -1 39 24 17 42 29 23 30 -1 24 11 -1 7 46 39 41 10 26 12 45 25 30 5 48 16 43 15 -1 17 -1 15 40 20 4 44 27 5 18 25 0 21 45 34 29 17 13 48 36 26 -1 12 -1 11 14 48 37 6 1 4 7 2 41 9 45 3 19 24 49 38 34 5 10 29 43 -1 -1 13 -1 2 23 -1 2 7 4 15 11 46 -1 14 24 5 48 47 3 31 11 8 20 15 33 4 43 27 6 1 30 7 29 -1 44 -1 20 13 10 -1 48 32 46 36 33 29 42 41 0 -1 24 44 26 30 39 27 -1 16 30 26 20 31 45 19 42 15 33 11 35 32 29 43 7 34 38 -1 10 0 24 42 9 46 12 11 44 8 34 26 38 15 31 -1 19 38 18 28 21 -1 7 49 26 35 24 32 36 42 21 39 33 13 48 41 43 10 9 15 12 29 4 20 28 14 2 44 18 5 47 31 11 6 19 8 22 -1 -1 14 1 37 39 18 41 -1 17 39 23 11 9 47 7 41 6 16 8 10 19 48 0 22 -1 42 4 48 1 0 34 7 5 39 20 44 21 16 49 12 33 43 3 14 37 47 23 35 17 28 13 45 27 24 31 9 19 18 30 25 26 6 2 15 22 11 -1 20 8 41 44 24 26 39 45 48 -1 17 29 16 9 46 2 5 -1 39 43 8 9 37 11 34 28 42 10 48 4 41 29 19 3 25 1 17 35 38 23 27 24 36 16 7 14 46 40 26 33 -1 10 1 3 14 11 8 32 21 9 22 46 47 4 29 42 23 25 -1 42 32 29 18 44 49 45 13 48 -1 1 43 39 10 3 18 48 22 47 -1 3 40 38 -1 46 22 24 30 12 0 9 17 39 35 -1 24 34 16 0 44 7 49 -1 41 14 45 44 35 -1 48 36 26 34 4 0 13 42 25 6 2 -1 30 -1 1 6 38 11 15 2 9 26 31 17 41 45 14 4 7 22 37 5 16 10 21 25 28 48 -1 31 35 44 6 4 16 -1 38 4 6 -1 9 30 38 24 3 29 -1 20 9 8 23 36 42 24 7 38 -1 8 49 30 20 4 21 17 32 -1 30 49 5 15 24 28 46 6 7 19 31 18 25 3 13 23 0 39 -1 37 18 -1				
Problem	36	25	30	5	2
	23 1 -1 24 15 -1 23 21 -1 10 12 -1 18 14 -1 7 22 19 6 23 12 -1 12 1 7 18 -1 -1 -1 8 -1 3 16 -1 -1 -1 -1 -1 -1 -1 8 -1 6 -1 -1 -1 18 8 12 15 9 -1 -1 -1 -1				
Problem	37	50	122	5	7
	-1 28 -1 -1 1 -1 28 13 17 24 10 40 11 -1 3 26 -1 3 18 -1 19 27 25 47 -1 13 14 25 -1 15 43 0 -1 -1 -1 39 3 -1 -1 22 47 -1 30 -1 -1 -1 20 1 21 -1 37 -1 -1 4 30 -1 4 2 24 -1 -1 3 -1 3 -1 -1 17 10 45 1 -1 49 26 -1 43 -1 45 -1 33 -1 26 37 -1 23 18 7 30 0 47 40 -1 30 26 -1 28 33 42 -1 15 -1 31 1 -1 40 24 2 42 19 -1 12 14 36 3 45 17 26 48 5 6 47 27 -1 12 38 43 28 0 14 29 26 8 6 10 48 -1 7 4 6				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	47 39 37 -1 29 48 32 19 7 21 44 27 2 -1 26 38 29 8 -1 7 38 6 10 19 -1 -1 43 -1 -1 -1 45 13 34 -1				
	39	25	60	10	4
Problem	16 12 4 11 9 -1 -1 20 18 -1 9 7 18 5 8 -1 -1 -1 21 -1 20 2 19 11 6 3 24 5 13 12 18 0 4 -1 3 24 0 1 16 6 23 4 -1 7 23 16 -1 17 18				
	9 8 -1 15 -1 3 10 -1 15 -1 20 -1 -1 -1 23 -1 -1 -1 -1 1 9 10 -1 9 19 12 4 1 -1 -1 2 22 16 0 6 -1				
	40	50	245	10	34
Problem	42 9 31 28 39 48 7 -1 -1 41 21 -1 49 17 35 31 40 29 0 7 11 45 24 -1 22 28 39 33 40 9 -1 -1 29 39 -1 -1 0 41 48 21 11 35 3 2 18				
	26 44 12 20 10 16 23 37 47 49 46 32 -1 -1 20 6 7 25 48 -1 7 45 37 9 33 44 8 39 26 46 -1 -1 -1 13 49 -1 -1 10 44 11 46 -1 14 23				
	13 -1 43 2 6 11 15 49 47 21 31 10 38 36 33 16 45 4 13 0 -1 13 32 15 16 43 7 17 -1 4 19 46 25 27 33 -1 19 31 0 40 -1 45 8 24 42				
	26 13 20 16 2 9 3 38 44 10 49 25 33 -1 20 -1 -1 22 20 -1 24 -1 1 -1 41 -1 16 23 9 21 18 39 30 -1 17 -1 -1 18 0 -1 6 44 23 38				
	27 28 43 29 4 37 20 35 7 15 8 24 3 -1 47 49 40 -1 15 46 29 10 48 43 -1 -1 24 38 -1 28 26 35 29 2 22 32 30 -1 35 15 33 -1 43 -1				
	29 36 30 11 44 1 45 -1 21 33 32 46 1 3 -1 23 41 4 13 28 26 38 12 5 14 33 8 -1 5 20 38 33 -1 28 47 22 6 2 41 21 32 42 17 -1 32				
	15 19 12 21 28 17 9 -1 22 41 48 29 38 46 0 31 12 2 -1 11 37 49 20 4 2 -1 20 -1				
	42	25	120	20	16
Problem	7 -1 12 13 0 23 4 6 7 9 2 -1 16 -1 13 16 19 17 5 24 14 23 21 10 6 7 0 1 8 11 4 20 18 15 -1 20 22 5 3 14 17 16 19 10 18 1 15 12				
	8 0 9 -1 20 7 24 -1 -1 16 8 13 23 4 -1 2 6 1 22 -1 13 20 7 24 21 0 17 23 -1 -1 23 -1 11 17 4 1 7 19 16 8 6 9 22 21 24 -1 3 -1				
	8 -1 -1 -1 6 8 0 24 9 1 -1 11 23 -1 16 21 0 18 2 -1 1 -1 7 0 14 23 4 8 9 16 2 12 -1 12 1 5 15 16 13 -1 12 5 -1 0 1 11 18 14 -1				
	43	50	490	20	94
Problem	-1 25 30 33 42 27 4 49 48 11 45 8 -1 40 35 27 9 4 3 46 7 12 47 1 8 25 31 24 33 -1 -1 43 28 21 41 15 29 40 39 32 -1 4 23 47 -1				
	31 21 25 20 -1 6 9 8 18 36 1 38 26 49 42 20 48 19 27 10 11 14 40 -1 -1 18 25 40 14 36 35 44 12 6 15 -1 37 31 -1 17 16 43 47 41				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	1 22 15 20 10 14 34 3 13 44 -1 43 32 40 16 31 5 29 25 33 14 4 22 7 49 36 10 26 19 -1 6 -1 15 10 19 36 -1 31 0 4 5 11 23 3 41 7 26 48 36 24 25 9 33 12 47 43 35 29 -1 -1 29 33 26 -1 0 45 19 36 16 48 -1 14 37 -1 39 40 29 44 13 6 8 21 49 38 31 9 2 11 17 12 22 27 30 0 24 7 34 36 26 32 28 -1 2 38 47 27 34 18 11 12 13 45 36 15 31 -1 49 14 43 45 32 48 20 34 16 23 42 44 -1 9 16 -1 30 47 17 -1 -1 -1 32 33 17 8 35 38 45 19 23 31 43 36 10 7 47 39 21 20 6 30 16 -1 42 17 14 38 16 33 15 -1 33 30 -1 38 1 41 37 22 7 14 23 20 19 13 -1 28 34 -1 26 12 -1 42 36 40 12 27 39 41 -1 3 25 2 -1 15 7 44 30 42 43 38 6 46 -1 26 -1 -1 3 2 6 15 37 25 22 33 34 20 9 13 21 14 -1 0 5 4 30 43 35 -1 24 7 41 37 -1 6 31 26 15 9 1 5 4 10 3 39 29 46 34 25 23 21 20 30 -1 0 24 35 30 16 43 34 10 14 46 39 37 27 18 6 1 -1 40 32 10 7 9 21 1 25 39 18 24 38 16 2 22 31 46 14 28 42 23 27 26 48 15 3 30 0 11 33 5 49 20 29 41 -1 35 45 15 0 -1 35 26 44 15 14 16 21 42 38 13 19 46 31 30 43 9 18 37 47 4 22 0 25 -1 20 24 35 8 11 37 19 7 33 25 39 2 31 36 32 38 4 26 41 18 34 48 16 -1 15 23 6 13 45 44 21 19 22 10 7 34 36 43 11 28 8 42 49 -1 15 16 12 10 41 49 33 29 27 32 5 40 18 46 36 7 42 1 28 6 19 26 -1 43 34 25 19 33 17 36 3 13 11 18 8 39 41 5 2 46 35 24 29 47 4 10 32 6 20 26 12 38 15 40 48 45 21 23 1 22 44 37 16 -1				
	46	15	11	5	2
Problem	-1 13 -1 -1 -1 -1 13 -1 5 7 -1 6 9 10 -1 -1 -1 -1 -1 13 -1 2 5 11 -1 -1				
	47	15	11	5	1
Problem	6 -1 -1 -1 7 -1 -1 1 4 -1 -1 10 -1 -1 -1 7 -1 8 9 -1 -1 5 -1 8 9 -1				
	48	15	11	5	1
Problem	-1 14 -1 5 -1 2 -1 -1 2 -1 -1 9 -1 10 -1 -1 3 5 7 -1 3 -1 -1 2 -1 -1				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	49	15	11	5	1
Problem	-1 -1 -1 -1 -1 8 14 -1 3 14 -1 12 -1 13 14 -1 0 3 -1 -1 -1 7 -1 9 -1 -1				
	50	15	11	5	0
Problem	3 9 -1 -1 -1 -1 -1 -1 2 -1 10 -1 0 1 4 -1 -1 11 -1 3 -1 11 -1 -1 3 -1				
	51	15	21	10	0
Problem	6 11 -1 -1 6 9 -1 -1 1 3 14 -1 2 6 11 12 -1 7 -1 -1 0 12 -1 12 -1 8 -1 1 2 -1 1 -1 -1 3 13 -1				
	52	15	21	10	2
Problem	8 -1 12 -1 1 14 -1 -1 1 9 -1 6 12 -1 -1 0 12 -1 0 -1 1 3 -1 1 13 -1 -1 1 2 11 -1 7 14 -1 9 -1				
	53	15	21	10	1
Problem	1 14 -1 8 13 -1 -1 4 13 -1 1 7 9 -1 2 13 -1 13 -1 8 -1 -1 2 7 -1 -1 6 -1 1 4 13 -1 4 -1 6 -1				
	54	15	21	10	2
Problem	7 -1 2 5 11 -1 3 8 9 -1 6 7 -1 5 -1 1 -1 -1 -1 6 10 -1 6 -1 1 12 13 -1 2 -1 1 8 -1 6 -1 -1				
	55	15	21	10	1
Problem	-1 2 6 11 12 -1 14 -1 1 6 10 -1 3 9 10 -1 13 -1 5 -1 3 13 -1 3 -1 0 -1 -1 6 -1 -1 2 4 -1 11 -1				
	56	15	42	20	4
Problem	7 13 -1 -1 6 8 13 -1 1 5 6 11 -1 0 6 13 -1 6 -1 -1 1 9 12 13 14 -1 1 2 3 10 -1 3 6 8 -1 0 1 5 9 13 -1 1 2 7 9 -1 1 8 10 -1 1 5 8 -1 8 12 -1				

Table 38: Details of DSM Problems Used for Static Testing

	Problem ID	Number of Elements	Number of Links	Percent of Total Links Active	Target Number of Feedbacks
	57	15	42	20	4
Problem	5 7 14 -1 0 -1 4 -1 1 2 6 -1 2 3 6 -1 9 -1 13 14 -1 3 10 -1 1 4 -1 0 2 3 4 6 8 12 13 14 -1 1 2 5 -1 0 2 6 14 -1 0 3 8 10 11 13 -1 3 -1 3 -1				
	58	15	42	20	6
Problem	4 7 8 11 -1 5 6 14 -1 0 1 10 -1 1 8 14 -1 0 1 3 9 14 -1 0 7 8 11 -1 2 7 8 -1 6 11 -1 -1 2 14 -1 5 11 12 14 -1 2 13 -1 11 -1 5 6 7 9 10 -1 4 -1				
	59	15	42	20	4
Problem	11 -1 2 3 5 7 10 -1 3 4 8 -1 0 -1 7 -1 3 12 -1 0 10 -1 -1 0 4 6 7 -1 0 1 4 8 -1 3 4 5 7 8 14 -1 2 9 -1 3 4 10 -1 1 3 4 5 8 11 -1 1 3 -1				
	60	15	42	20	10
Problem	1 2 7 10 11 -1 5 8 12 -1 1 5 6 -1 5 -1 0 8 9 11 14 -1 0 1 9 -1 1 -1 3 4 -1 6 10 13 -1 1 5 6 -1 0 1 4 5 9 -1 0 12 -1 0 3 10 -1 4 -1 3 12 -1				

APPENDIX C

PUBLICATIONS

C.1 Published Work

C.1.1 Thesis Relevant Conference Papers

1. Otero, R.E. and Braun, R.D.; “Calculating Data Importance using Mutual Information for Engineering Design,” AIAA 2010-9323, 13th AIAA ATIO/ISSMO Conference, Fort Worth, TX, September 2010.

2. Otero, R.E. and Braun, R.D.; “The Planetary Entry Systems Synthesis Tool (PESST): A Conceptual Design and Analysis Tool for EDL Systems,” 2010 IEEE Aerospace Conference, Big Sky, MT, March 2010.

3. Otero, R.E. and Braun, R.D.; “The State of Problem Decomposition in Engineering Design,” AIAA 2009-2188, 5th AIAA Multidisciplinary Design Optimization Specialist Conference, Palm Springs, CA, May 2009.

4. Otero, R.E.; Grant, M.J.; Steinfeldt, B.A.; and Braun, R.D.; “Introducing PESST: A Conceptual Design and Analysis Tool for Unguided/Guided EDL Systems,” 6th International Planetary Probe Workshop, Atlanta, GA, June 2008.

C.1.2 Other Conference Papers

1. Norton, C.D.; Fang, H.; Michel, T.; Moussessian, A.; Schiermeier, J.; Springer, P.; and Otero, R.E.; ”Model-Based Verification and Validation of Component Structures for RF and Optical Experimental Systems,” 2008 IEEE Aerospace Conference, Big Sky, MT, March 2008.

2. Joyner, R.; Osburg J.; Lentati, A.; Cole, B.; Otero R.E.; ”Evolving Conceptual Propulsion Design Using Preliminary Multidisciplinary Design Analysis and Optimization” 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Sacramento, CA, July 2006.

C.2 Planned Work

1. A journal article that combines the static decomposition and optimizer comparison work between Genetic Algorithms and MIMIC. This will also include results from the dynamic decision tree generation. Paper will be targeted to an MDAO journal.

REFERENCES

[1] Altus, S. S., Kroo, I. M., and Gage, P. J., "A genetic algorithm for scheduling and decomposition of multidisciplinary design problems," *Journal of Mechanical Design*, vol. 118, pp. 486–489, 1996.

[2] Anderson, J. D., *Hypersonic and High Temperature Gas Dynamics.* AIAA, August 2000.

[3] Awate, S. P., *Adaptive, Nonparametric Markov Models and Information-Theoretic Methods for Image Restoration and Segmentation.* PhD thesis, University of Utah, December 2006.

[4] Baluja, S. and Caruana, R., "Removing the genetics from the standard genetic algorithm," in *12th International Conference on Machine Learning*, Morgan Kaufmann Publishers, 1995.

[5] Baluja, S. and Davies, S., "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," in *Proceedings of the International Conference on Machine Learning*, pp. 30–38, 1997.

[6] Biernes, H. J., *Advances in Econometrics: Fifth World Congress, Volume 1.* Cambridge University Press, 1987.

[7] Bishop, C. M., *Pattern Recognition and Machine Learning.* Springer Science and Business Media, LLC, 2006.

[8] Bonet, J. D., Isbell, C., and Viola, P., "Mimic: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 424–430, 1997.

[9] Botev, Z. I., Grotowski, J. F., and Kroese, D. P., "Kernel density estimation via diffusion." To appear in Annals of Statistics., 2010.

[10] Braun, R. D., *Collaborative Optimization: An Architecture for Large-scale Distributed Design.* PhD thesis, Stanford University, May 1996.

[11] Braun, R. D., Gage, P., Kroo, I., and Sobieski, I., "Implementation and performance issues in collaborative optimization," in *Proceedings of the Sixth AIAA / NASA / ISSMO Symposium on Multidisciplinary Analysis and Optimization*, (Bellevue, WA), Sept 1996.

[12] Braun, R. D. and Manning, R. M., "Mars exploration entry, descent, and landing challenges," *Journal of Spacecraft and Rockets*, vol. 44, pp. 310–323, March-April 2007.

[13] Braun, R. D., Powell, R., Lepsch, R., Stanley, D., and Kroo, I., "Comparison of two multidisciplinary optimization strategies for launch vehicle design," *Journal of Spacecraft and Rockets*, vol. 32, pp. 404–410, May - June 1995.

[14] Brown, N., "Evaluation of multidisciplinary optimization (mdo) techniques applied to a reusable launch vehicle," Master's thesis, Georgia Institute of Technology, 2004.

[15] Browning, T. R., "Applying the design structure matrix to system decomposition and integration problems: A review and new directions," *IEEE Transactions on Engineering Management*, vol. 48, pp. 292–306, 2001.

[16] Chen, L. and Li, S., "Analysis of decomposability and complexity for design problems in the context of decomposition," *Journal of Mechanical Design*, vol. 127, pp. 545–557, 2005.

[17] Cho, S.-H. and Eppinger, S. D., "A simulation-based process model for managing complex design projects," *IEEE Transactions on Engineering Management*, vol. 52, pp. 316–328, August 2005.

[18] Chow, C. K. and Liu, C. N., "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, May 1968.

[19] Christian, J. A., Verges, A. M., and Braun, R. D., "Statistical reconstruction of mars entry, descent, and landing trajectories and atmospheric profiles," in *AIAA SPACE 2007 Conference & Exposition*, (Long Beach, CA), 18-20 September 2007. AIAA 2007-6192.

[20] Cybenko, G., "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, December 1989.

[21] Davis, D., Ihaka, R., and Fenstermacher, P., "Cryptographic randomness from air turbulence in disk drives," *Lecture Notes in Computer Science*, vol. 0839, pp. 114–120, 1994.

[22] Dec, J. A. and Braun, R. D., "Ablative thermal response analysis using the finite element method," in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, January 2009. AIAA 2009-259.

[23] DeJong, K. A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[24] D'Souza, C., "An optimal guidance law for planetary landing," in *AIAA*, 1997. AIAA 97-3709.

[25] Duda, R., Hart, P., and Stork, D., *Pattern Classification.* Wiley, 2001.

[26] Ford, J. and Bloebaum, C., "A decomposition method for concurrent design of mixed discrete/continuous systems," *Advances in Design Automation, Proceedings, 19th ASME Design Automation Conference*, vol. 65, no. 2, 1993.

[27] Fruchterman, T. M. J. and Reingold, E. M., "Graph drawing by force-directed placement," *Software - Practice and Experience*, vol. 21, pp. 1129–1164, November 1991.

[28] Gage, P., *New Approaches to Optimization in Aerospace Conceptual Design.* PhD thesis, Stanford University, 1995.

[29] Garey, M. R., Johnson, D. S., and Stockmeyer, L. J., "Some simplified np-complete graph problems," *Theoretical Computer Science 1*, vol. 3, pp. 237–267, 1976.

[30] Gen, M. and Cheng, R., *Genetic Algorithms & Engineering Design.* John Wiley & Sons, Inc., 1997.

[31] Goldberg, D. E., *Genetic Algorithms in Search Optimization and Machine Learning.* Addison Wesley, 1989.

[32] Grant, M. J. and Braun, R. D., "Analytic hypersonic aerodynamics for conceptual design of entry vehicles," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[33] Hajela, P., Bloebaum, C. L., and Sobieszczanski-Sobieski, J., "Application of global sensitivity equations in multidisciplinary aircraft synthesis," *Journal of Aircraft*, vol. 27, no. 1, pp. 1002–1010, 1990.

[34] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H., "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[35] Harish, P., Vineet, V., and Narayanan, P. J., "Large graph algorithms for massively multithreaded architectures," tech. rep., International Institute of Information Technology Hyderabad, 2009.

[36] Hart, W. E. and Belew, R. K., "Optimizing an arbitrary function is hard for the genetic algorithm," *Proceedings of the Fourth International Conference on Genetic Algorithms*, vol. 1, pp. 190–195, 1991.

[37] Haykin, S., *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2 ed., 1998.

[38] Holland, J. H., "Genetic algorithms: Computer programs that "evolve" in ways that resemble natural selection can solve complex problems ever their creators do not fully understand," in *Scientific American*, Nature Publishing Group, 1992.

[39] Hong, X., "Parzen windows." www.personal.reading.ac.uk/ ~sis01xh/ teaching/ CY2D2/ Pattern2.pdf.

[40] Humble, R. W., Henry, G. N., and Larson, W. J., *Space Propulsion Analysis and Design.* McGraw-Hill, 1995.

[41] Isbell, C. L., "Randomized local search as successive estimation of probability densities." A longer tutorial version of the 1997 paper on MIMIC that includes a derivation for MIMIC with trees. Can be accessed at http://www.cc.gatech.edu/ ~isbell/ tutorials/ mimic-tutorial.pdf.

[42] Ishikawa, M. and Yoshino, K., "Automatic task decomposition in modular networks by structural learning with forgetting," in *Proceedings of 1993 International Joint Conference on Neural Networks*, pp. 1345–1348, 1993.

[43] Jones, D., "Good practice in (pseudo) random number generation for bioinformatics applications," tech. rep., UCL Bioinformatics Group, 2010.

[44] Justh, H. L., Justus, C. G., and Keller, V. W., "Global reference atmospheric models, including thermospheres, for mars, venus and earth," in *AIAA / AAS Astrodynamics Specialist Conference and Exhibit*, (Keystone, Colorado), August 2006. AIAA 2006-6394.

[45] Keyhani, M., "Verification of thermal analysis codes for modeling solid rocket nozzles," NASA Contractor Report NASA-CR-195248, The University of Tennessee, May 1993. Unclassified, No Copyright, Unlimited, Publicly available.

[46] Khare, V. R., *Automatic Problem Decomposition using Co-Evolution and Modular Neural Networks.* PhD thesis, University of Brimingham, Birmingham, UK, 2006.

[47] Kipp, D. M., Dec, J. A., Wells, G. W., and Braun, R. D., "Development of a planetary entry system synthesis tool for conceptual design and analysis," in *Proceedings of the 3rd International Planetary Probe Workshop*, (Athens, Greece), June 2005.

[48] Klepikov, I. A., Katorgin, B. I., and Chvanov, V. K., "The new generation of rocket engines, operating by ecologically safe propellant "liquid oxygen and liquefied natural gas (methane)"," *Acta Astronautica*, vol. 41, no. 4, pp. 209–217, 1997.

[49] Knacke, T. W., *Parachute Recovery Systems Design Manual.* Para Pub, 1992.

[50] Kodiyalam, S. and Sobieszczanski-Sobieski, J., "Multidisciplinary design optimization - some formal methods, framework requirements, and application to vehicle design," *International Journal of Vehicle Design*, vol. 25, pp. 3–22, 2001.

[51] Kullback, S. and Leibler, R. A., "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[52] Kwak, N. and Choi, C.-H., "Input feature selection by mutual information based on parzen window," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1667–1671, December 2002.

[53] Lantoine, G., *A Methodology for Robust Optimization of Low-Thrust Trajectories in Multi-Body Environments.* PhD thesis, Georgia Institute of Technology, 2010.

[54] Laub, B., "Ablative thermal protection: An overview," in *55th Pacific Coast Regional and Basic Science Division Fall Meeting*, October 2003. Unclassified, No Copyright, Unlimited.

[55] L'Ecuyer, P., "Random number generation," tech. rep., Department d'Informatique et de Recherche Operationnelle, Universite de Montreal, 2004.

[56] Lees, L., "Laminar heat transfer over blunt-nosed bodies at hypersonic flight speeds," *Jet Propulsion*, pp. 259–269, 1956.

[57] Li, W., "Mutual information functions versus correlation functions," *Journal of Statistical Physics*, vol. 60, no. 5-6, pp. 823–837, 1990.

[58] Luke, S., Panait, L., Bassett, J., Hubley, R., Balan, C., and Chircop, A., "Ecj: A java-based evolutionary computation and genetic programming research system," 2002. Version: 18, http://www.cs.gmu.edu/ eclab/projects/ecj/.

[59] MacCormack, A., Rusnak, J., and Baldwin, C., "Exploring the structure of complex software designs: An empirical study of open source and proprietary code," in *Harvard Business School Working Paper 05-016*, 2005.

[60] Marsaglia, G., "Diehard battery of tests of randomness." Electronic, 1995.

[61] Matsumoto, M. and Nishimura, T., "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, pp. 3–30, January 1998.

[62] McCord, K. R., "Managing the integration problem in concurrent engineering," Master's thesis, Massachusetts Institute of Technology, May 1993.

[63] McLachlan, G. J., Do, K.-A., and Ambroise, C., *Analyzing microarray gene expression data.* Wiley-IEEE, 2004.

[64] Milos, F. S., Chen, Y.-K., Congdon, W. M., and Thornton, J. M., "Mars pathfinder entry temperature data, aerothermal heating, and heatshield material response," *Journal of Spacecraft and Rockets*, vol. 36, pp. 380–391, May-June 1999.

[65] Minzner, R. A., Reber, C. A., Jacchia, L. G., Huang, F. T., Cole, A. E., Kantor, A. J., Keneshea, T. J., Zimmerman, S. P., and Forbes, J. M., "Defining constants, equations, and abbreviated tables of the 1975 u.s. standard atmosphere," Tech. Rep. TR R-459, NASA, May 1976.

[66] Mitchell, T. M., *Machine Learning.* The McGraw-Hill Companies, Inc. and MIT Press, 1997.

[67] Mitcheltree, R. A., Moss, J. N., Cheatwood, F. M., Greene, F. A., and Braun, R. D., "Aerodynamics of the mars microprobe entry vehicles," *Journal of Spacecraft and Rockets*, vol. 36, pp. 392–398, May-June 1999.

[68] Moroz, V. I., "The atmosphere of venus," *Space Science Reviews*, vol. 29, pp. 3–127, March 1981.

[69] Morrisey, B. J., *Multidisciplinary Design Optimization of an Extreme Aspect Ratio HALE UAV.* PhD thesis, California Polytechnic State University, 2009.

[70] Olds, J., "System sensitivity analysis applied to the conceptual design of a dual-fuel rocket ssto," in *AIAA Paper*, (Panama City Beach, FL), September 1994.

[71] Otero, R. E. and Braun, R. D., "The planetary entry systems synthesis tool (pesst): A conceptual design and analysis tool for edl systems," in *IEEE Aerospace Conference*, (Big Sky, MT), March 2010.

[72] Page, W. A. and Woodward, H. T., "Radiative and convective heating during venus entry," *AIAA Journal*, vol. 10, pp. 1379–1381, 1972.

[73] Parzen, E., "On the estimation of a probability density function and the mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[74] Portree, D. S. F., "Mir hardware heritage," tech. rep., Johnson Space Center, March 1995. NASA RP-1357.

[75] Quinlan, J. R., *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann Publishers, 1992.

[76] Rasky, D. J. and Tran, H. K., "Low-cost entry systems for future planetary exploration missions," *Acta Astronautica*, vol. 45, pp. 347–355, August 1999.

[77] Regan, F. J. and Anandakrishnan, S. M., *Dynamics of Atmospheric Reentry.* AIAA, 1993.

[78] Rogers, J. L., "Demaid/ga - an enhanced design manager's aid for intelligent decomposition (genetic algorithms)," in *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pp. 1497–1504, September 1996. AIAA-96-4157.

[79] Rogers, J. L. and Bloebaum, C. L., "Ordering design tasks based on coupling strengths," tech. rep., NASA, July 1994. TM-109137.

[80] Rogers, J. L., "Tools and techniques for decomposing and managing complex design projects," *Journal of Aircraft*, vol. 36, pp. 266–274, January-February 1999.

[81] Rogers, J. L., McCulley, C. M., and Bloebaum, C. L., "Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes," Tech. Rep. 110247, NASA Technical Memorandum, April 1996.

[82] Rothlauf, F., *Representations for Genetic and Evolutionary Algorithms*. Springer, 2006.

[83] Schaeffer, S. E., "Graph clustering," *Computer Science Review*, vol. 1, pp. 27–64, 2007.

[84] Seiff, A., "Post-viking models for the structure of the summer atmosphere of mars," *Advances in Space Research*, vol. 2, no. 2, pp. 3–17, 1982.

[85] Seiff, A. and Kirk, D. B., "Structure of the venus mesosphere and lower thermosphere from measurements during entry of the pioneer venus probes," *Icarus*, vol. 49, pp. 49–70, January 1982.

[86] Seiff, A., Kirk, D. B., Young, R. E., Blanchard, R. C., Findlay, J. T., Kelly, G. M., and Sommer, S. C., "Measurements of thermal structure and thermal contrasts in the atmosphere of venus and related dynamical observations: Results from the four pioneer venus probes," *Journal of Geophysical Research*, vol. 85, pp. 7903–7933, December 1980.

[87] Shannon, C. E., "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, pp. 3–55, January 2001. Special issue reprinting Shannon's 1948 work with his corrections.

[88] Sobieszczanski-Sobieski, J., "Sensitivity of complex, internally coupled systems," *AIAA Journal*, vol. 28, no. 1, pp. 153–160, 1990.

[89] Srinivas, M. and Patnaik, L. M., "Genetic algorithms: A survey," *Computer*, vol. 27, pp. 17–26, June 1994.

[90] Steinfeldt, B. A., "Guidance, navigation, and control technology system trades for mars pinpoint landing," Master's thesis, Georgia Institute of Technology, May 2008.

[91] Steward, D. V., *Systems Analysis and Management: Structure, Strategy and Design*. Petrocelli Books, 1981.

[92] Sutton, K. and Graves, R. A., "A general stagnation point convective-heating equation for arbitrary gas mixtures," tech. rep., NASA, November 1971. TR R-376.

[93] SYSWERDA, G., "Uniform crossover in genetic algorithms," in *Proceedings 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishing, 1989.

[94] TAUBER, M. E. and SUTTON, K., "Stagnation-point radiative heating relations for earth and mars entries," *Journal of Spacecraft and Rockets*, vol. 28, pp. 40–42, Jan-Feb 1991.

[95] TODD, D., *Multiple Criteria Genetic Algorithms in Engineering Design and Operation.* PhD thesis, University of Newcastle, Tyne, UK, October 1997.

[96] TRAN, H. K., JOHNSON, C. E., RASKY, D. J., HUI, F. C. L., HSU, M.-T., CHEN, T., CHEN, Y. K., PARAGAS, D., and KOBAYASHI, L., "Phenolic impregnated carbon ablators (pica) as thermal protection systems for discovery missions," Tech. Rep. NASA TM-110440, NASA, April 1997. Unclassified, No Copyright, Unlimited, Publicly available.

[97] WAKAYAMA, S. and KROO, I., "Subsonic wing planform design using multidisciplinary optimization," *Journal of Aircraft*, vol. 32, no. 4, pp. 746–753, 1995.

[98] WHIFFEN, G. J., "Mystic : Implementation of the static dynamic optimal control algorithm for high-fidelity, low-thrust trajectory design," in *AIAA/AAS Astrodynamics Specialists Conference*, 2006.

[99] WHITFIELD, R. I., DUFFY, A. H. B., and GARTZIA-ETXABE, L. K., "Identifying and evaluating parallel design activities using the design structure matrix," in *International Conference on Engineering Design 2005*, (Melbourne, Australia), August 2005.

[100] WHITFIELD, R. I., SMITH, J. S., and DUFFY, A. H. B., "Identifying component modules," in *7th International Conference on Artificial Intelligence in Design (AID02)*, (Cambridge, UK), July 2002.

[101] WILLIAMS, S. D. and CURRY, D. M., "Thermal protection materials - thermophysical property data," tech. rep., NASA, December 1992. NASA RP-1289, Unclassified - Unlimited.

[102] WOLPERT, D. H., "No free lunch theorems for optimization," tech. rep., IBM Almaden Research Center, 1996.

[103] YU, T.-L., YASSINE, A., and GOLDBERG, D. E., "A genetic algorithm for developing modular product architectures," tech. rep., University of Illinois at Urbana-Champaign, October 2003.

[104] YU, T.-L., YASSINE, A., and GOLDBERG, D. E., "An information theoretic method for developing modular architectures using genetic algorithms," tech. rep., University of Illinois at Urbana-Champaign, April 2005. IlliGAL Report No. 2005014.

[105] Yu, T.-L., Yassine, A. A., and Goldberg, D. E., “An information theoretic method for developing modular architectures using genetic algorithms,” *Research in Engineering Design*, vol. 18, no. 2, pp. 91–109, 2007.