Calculating Data Importance using Mutual Information for Engineering Design

Richard E. Otero* and Robert D. Braun†

Georgia Institute of Technology, Atlanta, Georgia, 30332-0150, USA

As engineers examine larger coupled systems, computational complexity, available resources and the lack of expert intuition create a need to understand the importance of each link of data passed through an analysis. A better understanding and an automated calculation of this data importance would enable an advance of the art for automated decomposition and optimization methods. Larger coupled problems may, for instance, expand beyond an expert's experience in manual decomposition. By automatically discovering the importance and interrelated structure of a problem, low ranked data links might be temporarily separated to decompose a problem into sub-problems. A better understanding of the larger problem may also allow for organizational optimization around the coupled sub-problems discovered by this method, that is theoretically grounded in Information Theory.

Nomenclature

BC	Ballistic Coefficient
DSM	Design Structure Matrix
FPA	Flight Path Angle
GA	Genetic Algorithm
GSE	Global Sensitivity Equations
MSI	Module Strength Indicator
PESST	Planetary Entry Systems Synthesis Tool
Subscript	
i	Column index
j	Row index

I. Introduction

The scale of multidisciplinary problems in engineering has greatly increased over the past twenty years; fifty design variables once was typical of a large scale conceptual problem while now this number can exceed a thousand.[1] The processing power of computers has roughly followed the prediction of Moore for over thirty years, doubling by 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 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.

*Graduate Research Assistant, Daniel Guggenheim School of Aerospace Engineering, AIAA Student Member.

†David and Andrew Lewis Associate Professor of Space Technology, Daniel Guggenheim School of Aerospace Engineering, AIAA Fellow.

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.

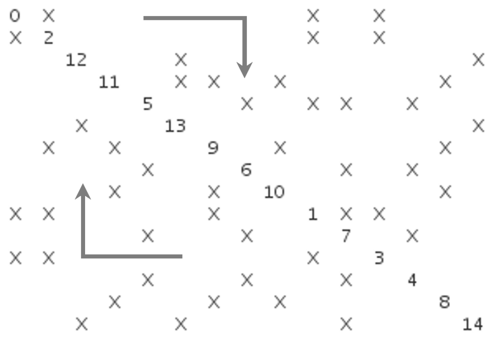
Decomposition methods expand the types of problems currently solvable by today's processing capabilities. A larger problem is decomposed into smaller, more tractable, sub-problems of appropriate structure. The solutions to these smaller problems are then used to either solve or provide insight into the behavior of the original problem. Historically this process has been performed by human field experts and many heuristics have been developed to guide these decomposition efforts.[2–6] 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. It is proposed that this creates a need for automated decomposition methods to aid the engineer in discovering useful structure that can be leveraged to understand and better solve their problems.

II. Background

II.A. Problem Representation

The most common problem representation used in multidisciplinary analysis and optimization (MDAO) for engineering design is the design structure matrix (DSM) or N^2 diagram.[7] The design structure matrix was introduced by Steward[8] as a tool for marking the interactions between analyses. This representation is commonly present in engineering classrooms and is used by most engineering decomposition methods for its ease of implementation and use.

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 1(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 1(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 would depend on the presence or strength of links connecting these sub-groups.


(a) Randomly Ordered DSM

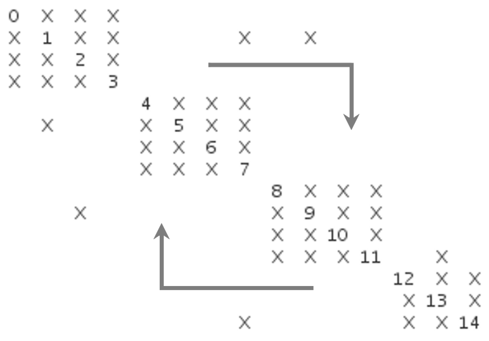

(b) Decomposed DSM

Figure 1. Two Orderings for the same DSM Problem Structure

II.B. 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. The four closely connected groups are considered as sub-problems in Figure 2(a). This assumes that the links between the groups are weak enough to temporarily 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 components of the problem. If links connecting the first two sub-groups in Figure 2(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 off of the importance of the links. A useful metric from information theory, mutual information, will be described along with its advantages over several current link heuristics in Section III.

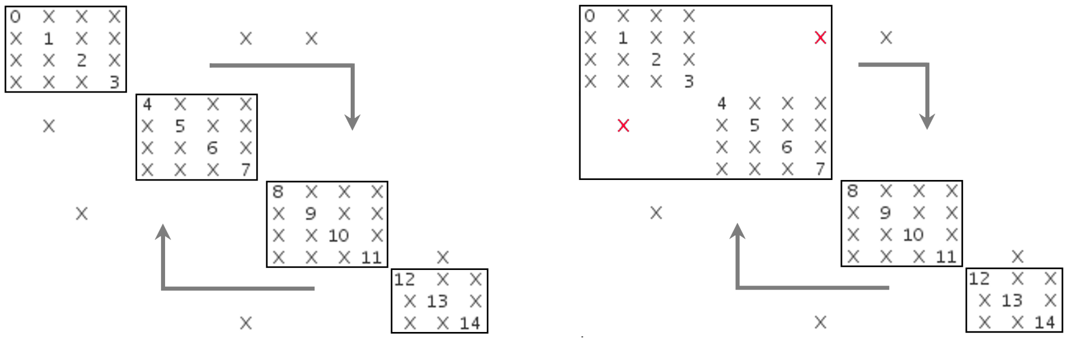

(a) Every Group formed into Sub-problems (b) Composite Sub-problem formed

Figure 2. 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[6,9] (binary connections), using heuristics such as the number of variables comprising the link,[1] and by using survey results from experts to rank linkage importance with a discrete value (ie. low/med/high).[4] A weighted DSM, with discrete values, ranks its links from a discrete set greater than two; integers 1-10, low/med/high,[4] etc. The information used to obtain this linkage ranking can come in the form of surveys or other means of expert supplied or computed data. Another metric used to assign a discrete value to a given linkage is the number of unique values being passed;[1] also known as the 'thickness of the pipe'. This would bias the importance measure toward links that transmit larger numbers of variables. Weighted DSMs can also have real values assigned to the links. These values have come from sensitivity calculations[3] or another metrics used to calculate a real valued importance for the link.[2,10]

Global Sensitivity Equations have been used at runtime to define the total derivatives of the output responses in terms of the subsystem local sensitivities.[3] This is one good instance where the dynamic importance of the links, given by the output response's sensitivity to changes, is computed at runtime. The method requires that derivatives can be taken, at both the subsystem and global level, but is one of the few methods that allows the problem behavior to dictate the importance that should be assigned to interdependencies.[11] This has seen use in the multidisciplinary synthesis of aircraft.[12]

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, greater reliance on high performance hardware will be required to handle future problem growth. Though the advancement of computational power is impressive, the opportunity of pairing this power with real-time decomposition methods should allow the efficient solution of large-scale problems.

II.C. Methods that Decompose Problems Pre/During Component Execution

The following decomposition methods have been separated by when they act to decompose the problem being examined. A class of methods seeks to use general metrics to reorganize the problem, before component execution, into closely coupled groups of components. 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 and these rankings can be leveraged to form clusters of related analyses. The drawback with pre-execution arrangement is that the expert opinion and/or heuristics are assumed to be applicable to the particular problem being analyzed.

Heuristics continue to be widely used to reorder the analyses in a DSM before the running of any of the processes.[2–6] Table 1 is a representative list of the heuristics that have been used to automatically decompose problems in engineering.

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.	Steward (1981)[8]
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)[3]
Pre/During-execution mix based on GSE Sensitivities	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. Static Loop Heuristics.	Given estimates for each analysis cost, attempts to find the least costly configuration.	Rogers [DeMAID-GA] (1996)[13]
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).	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)	Todd (1997)[14]
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)[5]
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)[15]
During-execution	Neural Networks	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)[16]
During-execution	Discrete DSM	Basic GA	The use 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)[17]

III. Proposed 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 3. An engineer seeking separable sub-problems may wonder what arrangement would lead to the best set of 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 available during a decomposition process the more informed a sub-problem separation can become.

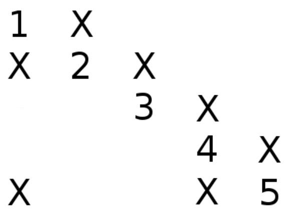
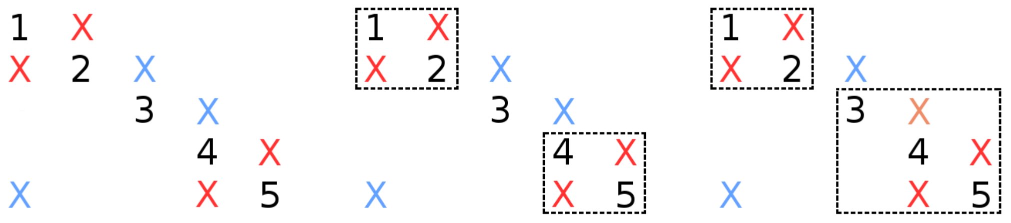


Figure 3. DSM Example with No Link Importance Information.

Adding link knowledge is much like adding color to the DSM in Figure 3. Two closely coupled problems are plainly seen with strong interior links in Figure 4(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 4(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 between groupings. 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 4. Importance of Link Ranking to Forming Sub-problem Clusters.

Applying this 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 be used to select links for temporarily removal, to better enable a decomposition; allowing the option to temporarily remove less important links from an analysis.

The current link metrics examined from Section II 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 decompose or learn more from it. 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 importances for those 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 partials 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.

III.A. 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.[18] It estimates the amount of shared uncertainty or dependence present between two random variables. This metric can be easily computed during the course of running a design structure matrix for each of the links in the 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 on continuous, discretized continuous and naturally discrete data.

Correlation is commonly used in engineering to measure dependence between random variables. Correlation though only measures the linear dependence between two variables while mutual information measures general dependence.[19] For example, two variables x and $y = sin(x)$ are uncorrelated but are correctly shown as having a high dependence when using mutual information. Correlated variables are always shown as dependent by mutual information, the dependence is then from a linear source.

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

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. 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 1 for continuous variables and in Equation 2 for discrete variables.

$$H(X) = -\int_X p(x)\log_2(p(x)) \tag{1}$$

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

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 for 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 Equation 2, 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 3.

$$\begin{aligned} H(X) &= -\sum_{x\in X} p(x)\log_2(p(x)) \\ &= -[0.25\log_2 0.25 + 0.25\log_2 0.25 + 0.25\log_2 0.25 + 0.25\log_2 0.25] \\ &= -[\log_2 0.25] \\ &= -[-2] = 2 \text{ bits minimum to describe option selected} \end{aligned} \tag{3}$$

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

$$H(X,Y) = -\int_X \int_Y p(x,y)\log_2(p(x,y)) \tag{4}$$

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2(p(x,y)) \tag{5}$$

Calculating the amount of mutual information also requires the calculation of the joint entropy which is the total sum of unique information provided by two random variables. In a Venn diagram, it would be the total area represented by both circles without double counting any overlap. This can be computed by Equation 4 for continuous variables or equation 5 for discrete variables.

$$I(A,B) = H(A) + H(B) - H(A,B) \tag{6}$$

Equation 6 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 $H(B)$, and the joint probability distribution between the variables, to calculate $H(A,B)$.

These probability distributions 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.

Mutual information as a metric also does not deal with the local behavior of the variable at one point. The value for mutual information is measured over the range of the probability distribution estimated by sampling from the space of solutions. This is 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.

III.B. Planetary Entry Systems Synthesis Tool

The Planetary Entry Systems Synthesis Tool (PESST) was developed by the authors to be a usable conceptual design tool for spacecraft entry studies of Earth, Mars and Venus.[20] The tool incorporates discipline models for geometry, hypersonic aerodynamics, guidance algorithms, trajectory simulation, thermal environment and sizing to converge conceptual entry vehicles.

PESST itself is a valuable contribution that allows conceptual designers to explore the space of potential entry scenarios. It serves to fill the nitch where a designer wishes to apply first-order physics to a conceptual design space to better understand the problem behavior. It will serve as a realistic example for using mutual information as a ranking metric in an engineering problem. A detailed discussion of the models utilized for the PESST tool is available from the cited reference.

IV. Results for Link Rank Metric

IV.A. Sample Problem

A design space was examined using the mutual information heuristic as a proof of concept for its use as a link metric. The PESST framework 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. 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).[21] 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.

The two strongest variables affecting skip out were shown to be entry FPA and entry velocity, their relationship to skip out over the domain is shown graphically in Figure 5. No cases were shown to skip out,

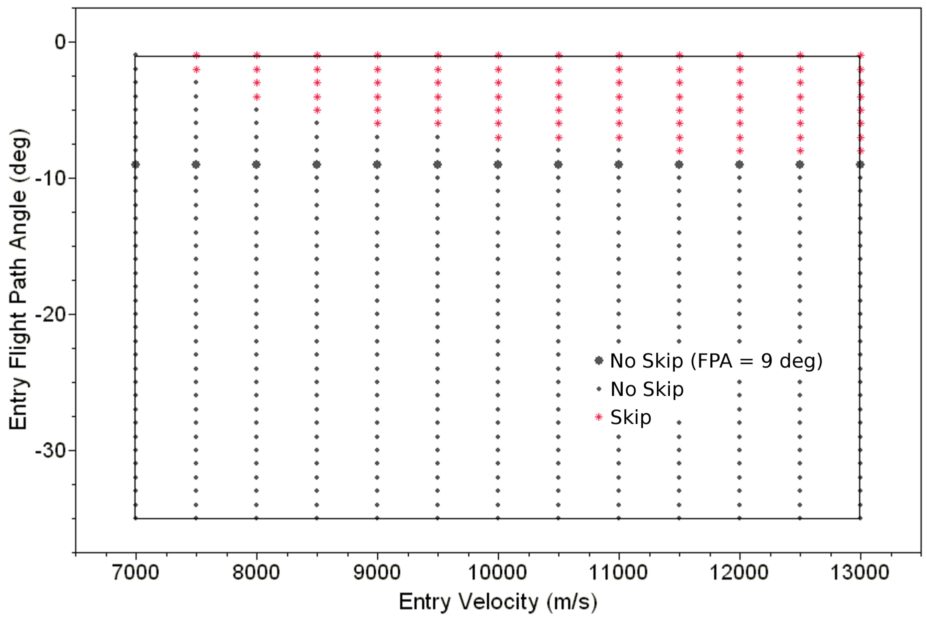


Figure 5. Design Space for a Venus Skip (BC = 200 kg/m^2).

for the domain examined, below negative nine degrees. Numerically Table 2 shows that entry FPA shared the greatest amount of information with the output variable for skip out.

Table 2. 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		

The input variables with more assigned choices had a higher entropy as 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 domain.

IV.B. Advantages for Ranking Link Importance

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 required by the analyses is significantly larger than the time 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 was on the order of seconds making it workable for this conceptual design tool.

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 covariance for linear dependence.

V. Static Decomposition with Mutual Information based Optimizer

Pre-execution (static) decomposition encompasses a very common set of methods used to automatically separate engineering problems into sub-problems. 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 or that they have values based from metrics that can be computed without evaluating the connected analysis tools. 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 links. The optimum arrangement for the problem is normally then searched 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.

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 multi-modal 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. By utilizing mutual information, the method is able to discover and leverage problem dependence structure.

V.A. MIMIC an Optimizer for Static Decomposition

Mutual Information Maximizing Input Clustering 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 design space by explicitly modeling the structural interactions between the input variables. 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 this model distribution for the solution space by using statistics from prior samples from the solution space.

Equation 7 describes the exact joint distribution between a set of N 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[22] uses pair-wise conditional probabilities to create an approximation to the true distribution. The probabilities are used to compute the mutual information between each pair of variables and a graph is formed with edges weighted by the mutual information between variables (represented by nodes). An example approximation for five random variables is shown in Equation 8. The selection of which conditional and marginal probabilities to use for the approximation is answered below.

$$p(X)_{true} = p(X_1|X_2...X_N)p(X_2|X_3...X_N)...p(X_{N-1}|X_N)P(X_N) \tag{7}$$

$$\hat{p}(X)_{approx} = p(X_4)p(X_5|X_4)p(X_1|X_4)p(X_3|X_4)p(X_2|X_3) \tag{8}$$

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.[23] Pair-wise conditional probabilities [ie $p(x \mid Y = y)$] are used to approximate the true joint distribution in Figure 7 as they are easier to determine from a smaller set of data. This is often a sufficient approximation to the joint distribution and was shown to lead to an order of magnitude reduction in function calls vs genetic algorithms, when tested on an example problem in Section VI.

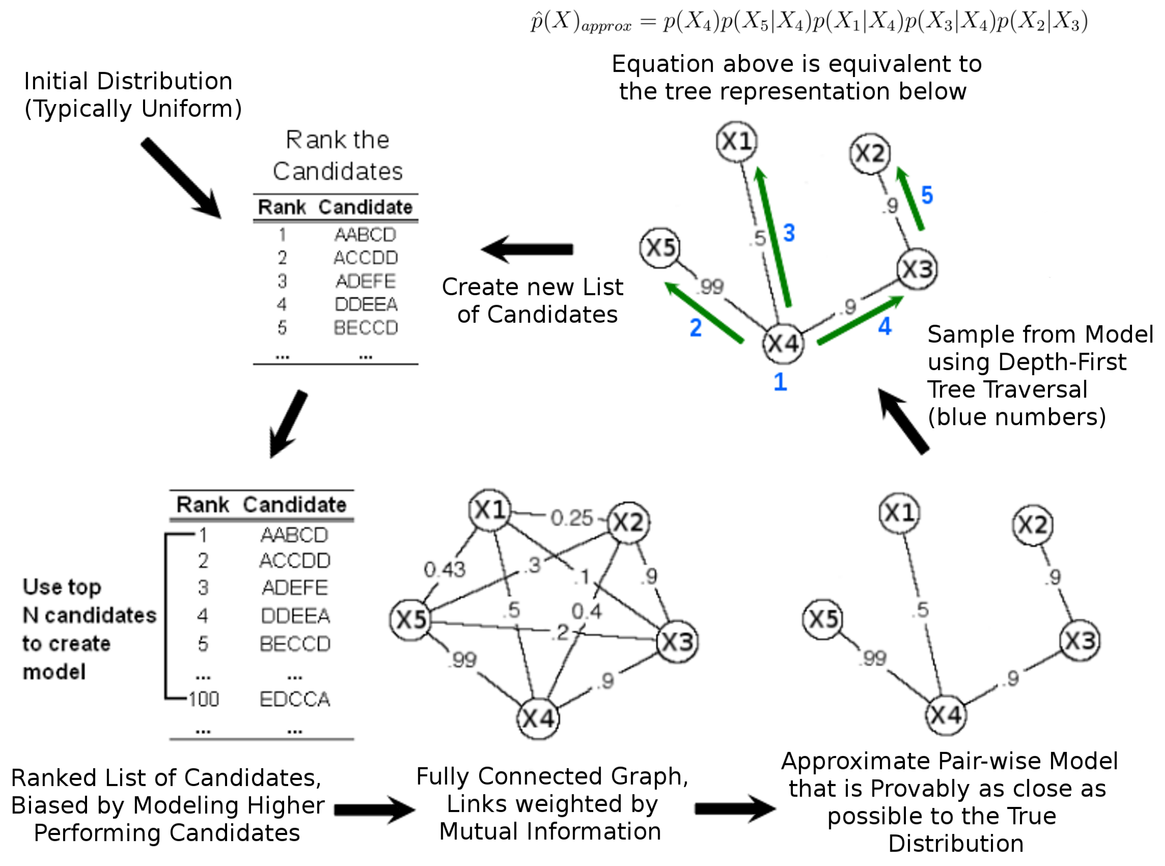

$\hat{p}(X)_{approx} = p(X_4)p(X_5|X_4)p(X_1|X_4)p(X_3|X_4)p(X_2|X_3)$
Equation above is equivalent to the tree representation below
Initial Distribution (Typically Uniform)
Rank the Candidates
Rank Candidate
1 AABCD
2 ACCDD
3 ADEFE
4 DDEEA
5 BECCD
... ...
Create new List of Candidates
Sample from Model using Depth-First Tree Traversal (blue numbers)
X1
X2
X3
X4
X5
.5
.9
.99
.9
1
2
3
4
5
Use top N candidates to create model
Rank Candidate
1 AABCD
2 ACCDD
3 ADEFE
4 DDEEA
5 BECCD
... ...
100 EDCCA
... ...
0.25
0.43
.3
.1
.9
.5
0.4
.2
.99
.9
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

Figure 6. Flow of execution for the MIMIC algorithm.

To show how the statistical data is used to create a distribution model, each variable is first assigned a vertex in the graph that will serve as the approximated model, the example five node graph in Figure 6. The mutual information between each pair of variables i and j is computed; this only requires knowledge of $p(x_i)$ and $p(x_i, x_j)$. The mutual information can be calculated from this information and used to weight the links between the variables. The fully-connected graph with computed weights is shown in figure 6 for a five variable 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.

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.[24] The model with pair-wise statistics that minimizes the KL distance between it and the true distribution will be the best possible approximation for the true distribution available.[25] The graph that minimizes the KL distance will be a maximum spanning tree with the edges weighted by mutual information. A discussion on how this occurs can be found in Chow[25] and Baluja.[22]

To form this maximum spanning tree, 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 tree. 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. The maximum spanning tree for the example is shown as the acyclic five node graph in Figure 6.

This tree model for the distribution is equivalent to the best possible approximation to the true distribution, using only single and pair-wise conditional probabilities. The tree shown in Figure 6 is equivalent to the distribution described by Equation 8.

Sampling from the tree, a model of the solution space, 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 estimated Equation 8 comes from the tree shown in Figure 6; 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, how can the model be used to sample from higher performing areas? This model is biased towards the higher performing areas of the solution space by using the lowest member of the top M percent of the data to specify the low-bar for data used to form the next generation's model. In MIMIC, M is typically 50 percent which marks the position for the group mean, in a ranked list. Further sampling from the old model is continued, to replace the lower than average candidates, until a new group of the same size is formed with samples that are all better than the past mean. This new group 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 6 for the full pictorial flow of the MIMIC algorithm.

V.B. Extension to MIMIC for Continuous Variables

The MIMIC algorithm requires a 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 III.A was shown to be possible with either continuous or discrete random variables. A dependence tree (as in MIMIC) can still be used to approximate the full joint distribution between all the input variables once the mutual information is known.[25] 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 that will utilized in this work is Parzen-window density estimation. Parzen's work[26] 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,[27] image restoration[28] and regression.[29]

$$p(\bar{x}) \approx k/NV = \underbrace{\frac{1}{Nh^d}\sum_{i=1}^{N} K\left(\frac{\bar{x}-\bar{x}_i}{h}\right)}_{\text{h is edge for cube of dimension d}} \tag{9}$$

The value k which represents the number of samples appearing within the region R is often calculated by applying a function at each sample point, Equation 9. The 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.[30]

$$K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x_i-x-h\mu_x)^2}{h^2\sigma_x^2} + \frac{(y_i-y-h\mu_y)^2}{h^2\sigma_y^2} - \frac{2\rho(x_i-x-h\mu_x)(y_i-y-h\mu_y)}{h^2\sigma_x\sigma_y}\right]\right) \quad \text{where } \bar{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{10}$$

Many kernel functions are available but the one selected for this work will be the Gaussian kernel. The selection of this kernel makes a weak assumption that the true distribution is smooth. Data distributions meetings this assumption will be 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 is available to any arbitrary pdf function.[31] 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.

$$K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) = \underbrace{\frac{1}{2\pi}\exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2h^2}\right)}_{\text{where } \mu_x=\mu_y=0,\ \sigma_x=\sigma_y=1,\ \rho=0} \tag{11}$$

The full kernel for a 2-D Gaussian is presented in Equation 10. The model is simplified by using standard Gaussian's with each $\mu = 0$ and a $\sigma = 1$ as in Equation 11. 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 12.

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[32] can be utilized to dynamically select for the parameter h.

$$\begin{aligned} p(\bar{x}) = p\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = p(x,y) &\approx \frac{1}{Nh^2}\sum_{i=1}^{N} K\left(\frac{\bar{x}-\bar{x}_i}{h}\right) \\ p(x,y) &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\pi\, h^2}\exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2h^2}\right) \end{aligned} \tag{12}$$

Parzen-window estimation for $p(x,y)$, using a Gaussian kernel, requires the use of Equation 12 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 13.

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{2\pi}\, h}\exp\left(-\frac{(x-x_i)^2}{2h^2}\right) \tag{13}$$

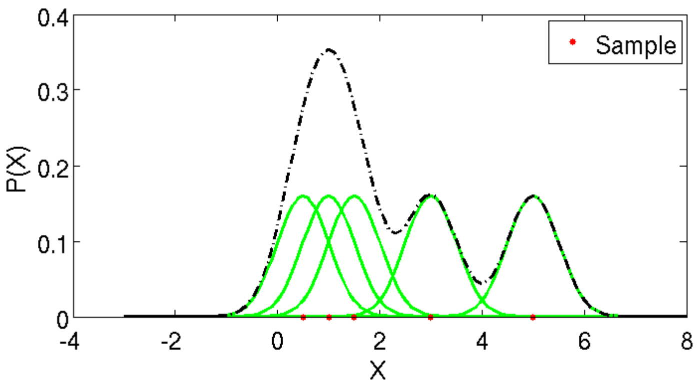


Figure 7. Approximation from Five Samples using Gaussian Kernel. (h=0.5)

This can be intuitively considered as a summation of Gaussians, placed at each sample point, that is then normalized to create a pdf estimate, Figure 7.

By using Parzen-window density estimation, the components needed by MIMIC for structural modeling can be found. This allows MIMIC to be applied to both continuous and discrete variables. The continuous and discrete versions of the mutual information equation would then be used to form a tree model for the algorithm. The use of kernel density estimation, specifically Parzen Window Estimation, to calculate probability densities allows for the estimation of mutual information between continuous variables.[29]

VI. Results for Static Decomposition Method

VI.A. Analytic Demonstration for Leveraging Problem Structure

An analytical example was used to show how MIMIC can discover the structure of a problem, leveraging it to converge onto a solution faster than genetic algorithms. 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 a proof of concept to the contention that MIMIC can provide an improvement to the current practice which utilizes genetic algorithms; when the problem displays structure that can be leveraged.

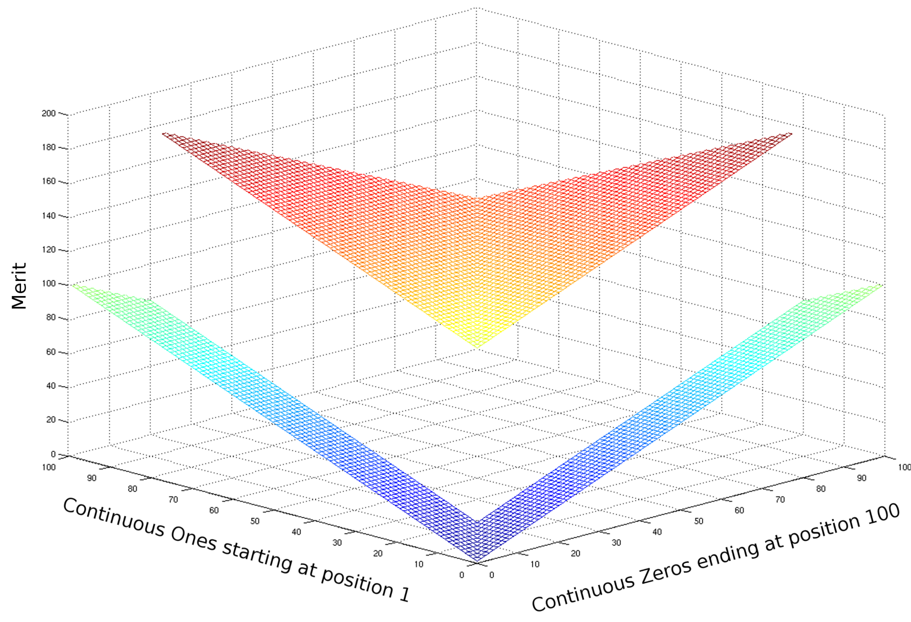


Figure 8. Four Peaks (Side View) for an input $\bar{X}$ of size 100, T=10.

The four peaks problem was used by Baluja to describe the behavior of PBIL[33] and later by DeBonet for MIMIC.[23] The problem has two local minima, a string of all 1s or all 0s, and two global minima; either N-(T+1) leading 1s, or trailing 0s, with the remaining T+1 of the opposite value, see Figure 8. 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 8. 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 a given cut off, there is a rewarded boost to the utility of the solution. Leading and trailing values are explained by example in Equation 14.

$$Example\ Candidate: \left[\underbrace{11111}_{\text{head=5}}\,010110111\,\underbrace{000000}_{\text{tail=6}}\right] \tag{14}$$

A mathematical description for the four peaks problem is shown in Equations 15. 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 14 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 $6 + N = 6 + 20 = 26$.

$$\bar{f}(\bar{X}, T) = max[tail(0, \bar{X}), head(1, \bar{X})] + reward(\bar{X}, T) \tag{15a}$$

$$tail(0, \bar{X}) \text{ equals the number of trailing 0s in } \bar{X} \tag{15b}$$

$$head(1, \bar{X}) \text{ equals the number of leading 1s in } \bar{X} \tag{15c}$$

$$reward(\bar{X}, T) = \begin{cases} N \text{ if } tail(0, \bar{X}) > T \text{ and } head(1, \bar{X}) > T \\ 0 \text{ otherwise} \end{cases} \tag{15d}$$

This example problem allows the toggling of the problem size, by changing the size of N, and the toggling 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 9 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.

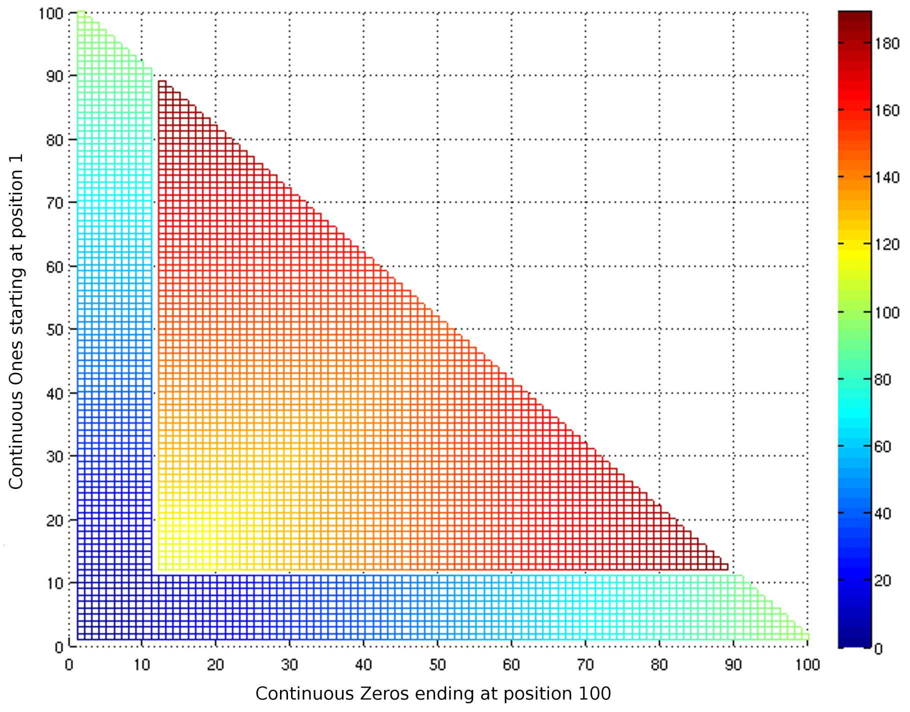


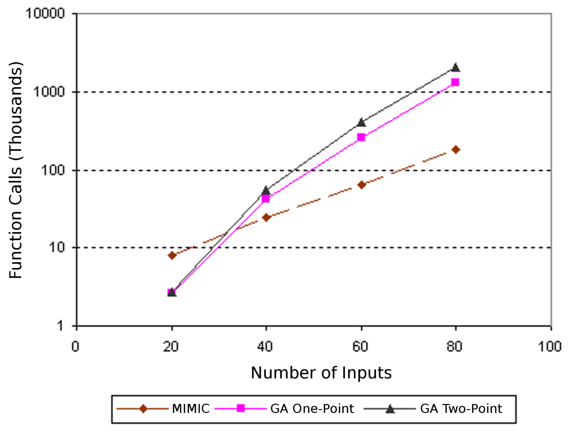
Figure 9. Four Peaks (Top View) for an input $\bar{X}$ of size 100, T=10.

A mature third-party implementation of genetic algorithms[34] 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.

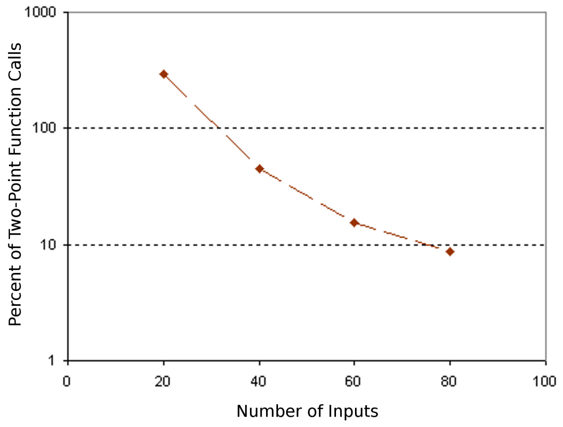
Genetic algorithms are often used in engineering for multimodal domains and to apply decomposition heuristics for coupled problems. Pair-wise interactions between variables are not explicitly modeled by the crossover operation which randomly separates and passes on candidate solutions. The crossover operation was created for the carrying forward of sub-problem solutions but Figure 10(a) shows the challenge two crossover operators (single and two-point crossover) have on the Four Peaks Problem. Explicitly modeling the pair-wise inter-dependencies between variables allows MIMIC to obtain an order of magnitude improvement when working with 80 inputs, Figure 10(b).

The points in the chart are bounded averages computed with 95% confidence, Table 3, for all three methods. 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



(a) Absolute Number



(b) Percentage of Two-Point Function Calls

Figure 10. Function Calls Required to Find the Global Optimum.

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 were required for 80 inputs as opposed to the 20667 iterations required to converge using two-point crossover, Table 3.

Table 3. 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’ always correctly placed at the first variable, aiding it to a potentially correct separation of the two components for 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 the small version of this problem. The pair-wise models become more useful as the number of variables increase and allows for solutions at a tenth the cost of the two-point crossover at 80 inputs. Even without the application of mutual information clustering to the ranking of links in a DSM, so many heuristics use GAs that MIMIC could cause a 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 the toggling of the problem size to measure the scalability for three methods (MIMIC, GA with One-Point, 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, in a DSM, 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 preconditioning the distribution. This quality of MIMIC has application to a wide variety of aerospace problems. Generally speaking, knowledge of the physics to 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.

VII. Conclusions

Mutual information is able to be used on continuous, discretized continuous or naturally discrete variables which are all commonly found in engineering problems making it very flexible as a metric. The metric also provides a measurement of dependence between variables beyond what is commonly found with covariance. This could be used as a ranking metric to compute the importance of links in a design structure matrix. The inputs to each contributing analysis could be clustered based on their mutual information leading to high quality sub-problem groupings for a system analysis that does not depend on manual decomposition.

A global search method, new to aerospace engineering, was shown that uses mutual information to leverage problem structure. This method gained up to an order of magnitude improvement in solution cost measured from the example problem used. This demonstrates a potential for MIMIC as a drop in replacement for genetic algorithms when applying heuristics to static decomposition problems. An extension was shown that would allow for the use of continuous variables with the MIMIC algorithm.

Acknowledgments

The authors wish to thank Charles L. Isbell, Jr., one of the original creators of the MIMIC algorithm, for his promotion of the algorithm within Computer Science. The PESST framework received excellent analysis contributions in guidance from Brad A. Steinfeldt and Michael J. Grant and in parametric analysis from Patrick J. Smith. The authors thank the support of the Jet Propulsion Laboratory and their insight and feedback throughout the development of PESST.

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, 1996, pp. 486–489.

[2]Ishikawa, M. and Yoshino, K., "Automatic Task Decomposition in Modular Networks by Structural Learning with Forgetting," *Proceedings of 1993 International Joint Conference on Neural Networks*, 1993, pp. 1345–1348.

[3]Rogers, J. L. and Bloebaum, C. L., "Ordering Design Tasks Based on Coupling Strengths," Tech. rep., NASA, July 1994, TM-109137.

[4]Rogers, J. L., "Tools and Techniques for Decomposing and Managing Complex Design Projects," *Journal of Aircraft*, Vol. 36, No. 1, January-February 1999, pp. 266–274.

[5]Whitfield, R. I., Smith, J. S., and Duffy, A. H. B., "Identifying Component Modules," *7th International Conference on Artificial Intelligence in Design (AID02)*, Cambridge, UK, July 2002.

[6]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.

[7]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, 2001, pp. 292–306.

[8]Steward, D. V., *Systems Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, 1981.

[9]Chen, L. and Li, S., "Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition," *Journal of Mechanical Design*, Vol. 127, 2005, pp. 545–557.

[10]Cho, S.-H. and Eppinger, S. D., "A Simulation-Based Process Model for Managing Complex Design Projects," *IEEE Transactions on Engineering Management*, Vol. 52, No. 3, August 2005, pp. 316–328.

[11]Sobieszczanski-Sobieski, J., "Sensitivity of Complex, Internally Coupled Systems," *AIAA Journal*, Vol. 28, No. 1, 1990, pp. 153–160.

[12]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, 1990, pp. 1002–1010.

[13]Rogers, J. L., "DeMAID/GA - An Enhanced Design Manager's Aid for Intelligent Decomposition (Genetic Algorithms)," *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, September 1996, pp. 1497–1504, AIAA-96-4157.

[14]Todd, D., *Multiple Criteria Genetic Algorithms in Engineering Design and Operation*, Ph.D. thesis, University of Newcastle, Tyne, UK, October 1997.

[15]Whitfield, R. I., Duffy, A. H. B., and Gartzia-Etxabe, L. K., "Identifying and Evaluating Parallel Design Activities using the Design Structure Matrix," *International Conference on Engineering Design 2005*, Melbourne, Australia, August 2005.

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

[17]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, 2007, pp. 91–109.

[18]Shannon, C. E., "A Mathematical Theory of Communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 5, No. 1, January 2001, pp. 3–55, Special issue reprinting Shannon's 1948 work with his corrections.

[19]Li, W., "Mutual Information Functions Versus Correlation Functions," *Journal of Statistical Physics*, Vol. 60, No. 5-6, 1990, pp. 823–837.

[20]Otero, R. E. and Braun, R. D., "The Planetary Entry Systems Synthesis Tool (PESST): A Conceptual Design and Analysis Tool for EDL Systems," *IEEE Aerospace Conference*, Big Sky, MT, March 2010.

[21]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, No. A13, December 1980, pp. 7903–7933.

[22]Baluja, S. and Davies, S., "Using optimal dependency-trees for combinatorial optimization: Learning the Structure of the Search Space," *Proceedings of the International Conference on Machine Learning*, 1997, pp. 30–38.

[23]Bonet, J. D., Isbell, C., and Viola, P., "MIMIC: Finding Optima by Estimating Probability Densities," *Advances in Neural Information Processing Systems (NIPS)*, 1997, pp. 424–430.

[24]Kullback, S. and Leibler, R. A., "On Information and Sufficiency," *The Annals of Mathematical Statistics*, Vol. 22, No. 1, 1951, pp. 79–86.

[25]Chow, C. K. and Liu, C. N., "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Transactions on Information Theory*, Vol. 14, No. 3, May 1968, pp. 462–467.

[26]Parzen, E., "On the Estimation of a Probability Density Function and the Mode," *Annals of Mathematical Statistics*, Vol. 33, 1962, pp. 1065–1076.

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

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

[29]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, No. 12, December 2002, pp. 1667–1671.

[30]Hong, X., "Parzen Windows," www.personal.reading.ac.uk/ ∼sis01xh/ teaching/ CY2D2/ Pattern2.pdf.

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

[32]Botev, Z. I., Grotowski, J. F., and Kroese, D. P., "Kernel Density Estimation Via Diffusion," To appear in Annals of Statistics.

[33]Baluja, S. and Caruana, R., "Removing the Genetics from the Standard Genetic Algorithm," *12th International Conference on Machine Learning*, Morgan Kaufmann Publishers, 1995.

[34]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/.