The State of Problem Decomposition in 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 for advancing the state of automated decomposition methods. Larger coupled problems may, for instance, expand beyond an expert's experience in manual decomposition. This paper describes the current state of automated decomposition within engineering and explores several promising ideas that could be utilized to advance the state of the practice. It is proposed that the explicit modeling of dependencies is both possible and available for a new avenue of advancement in how problem decomposition is applied to engineering.

Nomenclature

i	Row in DSM
j	Column in DSM
N	Number of binary inputs for the sample problem
T	Toggles the size of input space that receives a performance reward
DSM	Design Structure Matrix
EDA	Estimation of Distribution Algorithms
GA	Genetic Algorithm
GSE	Global Sensitivity Equations
KL	Kullback Leibler; a measure of 'distance' between two distributions
MDL	Minimum Description Length
MIMIC	Mutual Information Maximizing Input Clustering
ML	Machine Learning
MSI	Module Strength Indicator
NN	Neural Networks
PC	Probability Collectives
RSE	Response Surface Equation

I. Introduction

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, problems. The solutions to these smaller problems are then used to either solve or provide insight into the behavior of the original. Historically this process has been performed by humans (usually experts in their field) and many heuristics have been developed to guide these decomposition efforts.[1–5] 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.

[*]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.

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 one thousand. Automated decomposition could be used to more accurately learn the coupling (interdependency) between disciplines and could lead to large reductions in the time it takes to analyze problems by reducing the computational design space. These methods may also provide some insight into the physics driving the problem through different areas of the design space.

A literature review is provided for the current state of problem decomposition in engineering. Mutual Information Maximizing Input Clustering (MIMIC) will be described as a method for automatically decomposing coupled problems, pulling from important work that has been performed with Estimation of Distribution Algorithms (EDA). To demonstrate this method, an example coupled analytic problem is defined which allows one to test decomposition methods by toggling the size of the problem.

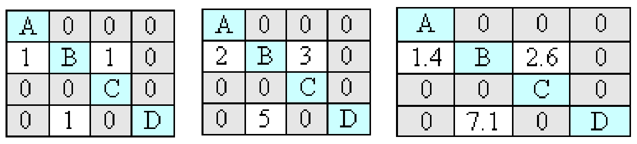
II. Literature Review of Decomposition in Engineering

When evaluating how to decompose a coupled problem into sub-problems, a great challenge has been in the correct evaluation of the importance due to each of the interconnections. A highly coupled problem with very weak connections could easily be treated as a decoupled problem (without interconnections); the connections being re-added only to refine the final answer. 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, possessing strongly interconnected dependencies, challenges efforts to separate it into smaller tractable pieces. Researchers have hoped to leverage knowledge of these dependencies, to gain the benefits of decomposition. A review of the methods currently used in engineering to rank and utilize these interconnections follows:

II.A. Design Structure Matrix Representations

The design structure matrix (DSM) was introduced by Steward[6] as a tool for describing the interactions between analyses. Most decomposition methods have utilized the DSM matrix representation for engineering problems because of its ease of implementation and wide presence in engineering classrooms. A DSM is only one of several possible representations for a problem and is only useful as a means to clarify the interactions between analyses. This paper will follow the standard DSM format where feed-back connections are above the diagonal while feed-forward connections are shown below the diagonal.

This paper will make a distinction between DSMs that are used in the literature, based upon the form of their linkage information. As seen in Figure 1, the three distinct groups are: Binary, Discrete, and Weighted DSMs.



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

(a) Binary DSM

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

(b) Discrete DSM

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

(c) Weighted DSM

Figure 1. Distinctions made between Design Structure Matrices.

Binary DSMs only contain on/off linkage information between analyses. Existing links are often treated as being equally important as there is no ranking information available.[7,8] Discrete DSMs rank their links from a discrete set greater than two; integers 1-10, low/med/high,[5] etc. The information used to obtain this linkage information 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;[9] also known as the 'thickness of the pipe'. This would bias the importance measure toward links that transmit larger numbers of variables. Weighted DSMs have real values assigned to the links. These values have come from sensitivity calculations[2] or another metrics used to calculate a real valued importance for the link.[1,10]

II.B. Implicit Statistical Treatments of Decomposition in Engineering

II.B.1. Decomposition with the Genetic Crossover Operation

Genetic Algorithms (GAs) are currently an accepted method for searching large multimodal domains that have inexpensive fitness function calls. The requirement regarding the execution time for the fitness function is needed as GAs often require several hundred thousand function calls to converge onto a solution. The method used in GAs, to model the solution space, is to maintain a population of candidate solutions. The biologically inspired operations of reproduction, crossover and mutation are performed on these solutions to improve their utility over successive iterations.

Input-variable dependencies, for a solution space, would 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 will appear near each other on the candidate string. This bias for the crossover operation, will be shown through the examples of this section.

The application of crossover within GA is an attempt to separate out a sub-solution to a component 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. 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.

In practice, expert knowledge is sometimes pulled upon to place related inputs 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 access to expert knowledge for the domain at hand and presupposes that the relationships for this domain will fall within previously observed behaviors.[11]

The following examples show the three main methods currently used by genetic algorithms to perform the crossover operation. The implicit modeling of coupling behavior is suggested as being one reason GAs require a large number of function calls for successful automatic problem decomposition.

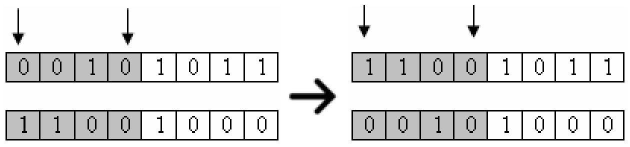


Figure 2. Single-point Crossover.

For a one-point crossover, the segment to crossover always starts at the first element. The second point that shows where the segment will end is determined randomly. The two sections are then switched between the two strings; this generates two children with the same number of elements as their parents, figure 2.

One-point crossover is unequal in how it treats the element positions in a candidate solution. By fixing the initial index at the first element, the first element will always be a member of the transferred segment. In the case where the sub-problem that the GA decomposes is composed from the first elements and the last elements, it would be impossible for this knowledge (found by the GA), in one parent, to be passed on to the next generation. Any re-creation of the solution would have to be formed of segments from separate parents. Without a method to explicitly model the interactions between inputs, the method relies on many random samples to eventually generate the needed solution.

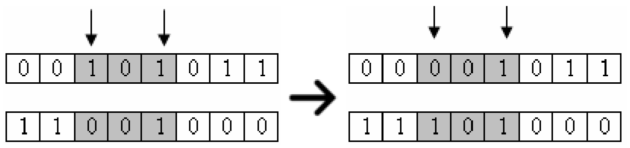


Figure 3. Two-point Crossover.

Two-point crossover increases the chances that candidates can be separated into their component sub-problems during a single generation. This can help a discovered decomposition pass on during successive generations. Both indexes are randomly selected and the points between them are swapped to generate two children, Figure 3. Using this method, a sub-problem composed of inputs from the start and end of the candidate string have the chance of being decomposed together, as one unit.

Though the representational power is 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 requires a large number of iterations.[12]

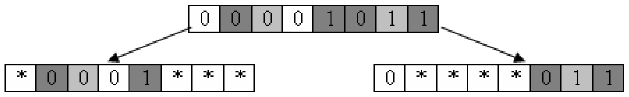


Figure 4. Two-point Crossover Performed on a Problem with Interleaved Sub-problems.

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 4. In Figure 4, the light grey inputs deal with the solution to one sub-problem while the dark gray inputs represent a second sub-problem. Randomly, at least two sets of proper splits must be done for the sub-problem solution to be separated and passed on.

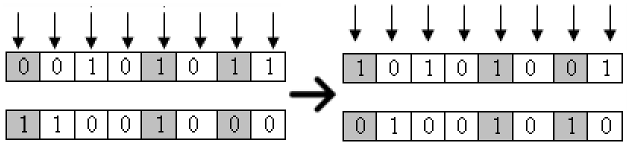


Figure 5. Uniform Crossover.

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. This means that each element has a chance of being traded between the parents; independent from the chance any of the other elements were traded. The highlighted sections in Figure 5 represent the elements that stochastically were selected for swapping. The light gray and dark grey highlighting in Figure 4 and Figure 6 represent two different input groupings that would each relate to a separable sub-problem, if they could be found.

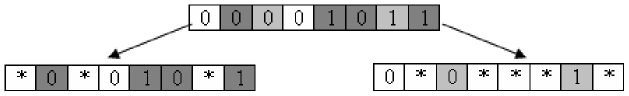


Figure 6. Uniform Crossover Performed on a Problem with Interleaved Sub-problems.

Uniform crossover is the most general in terms of its ability to separate sub-problems, as in Figure 6, but it also has the largest space of possible splits. The biological inspiration for genetic algorithms is often quoted and is easy for an audience to respond to. Its easily understood operations and the ease of implementation has made it one of the main methods taught and used by engineers over multi-modal solution spaces. Each of these three crossover operations is an unguided random selection that assumes, over many iterations, well performing patterns will stably remain within the population. This means that any increase in representational power for the expression of sub-problems is also hindered by an increase in the space of all possible splits that must be searched by the random process.

II.B.2. Treatment of Decomposition in Probability Collectives

Probability Collectives is a method of optimization that builds a model of the solution space through the behavior policies developed by several independent agents. Here, an agent is defined as an actor that is able to learn a policy that increases its own perceived reward. This measure of reward is directly tied to the global performance and indirectly tied to the actions of other agents. Here, interaction information between variables is handled indirectly. Though an agent could control several variables, work to date has one agent for every input variable in an already decoupled system. The system output value is used to guide each agent in what values they should choose. Although the idea has been suggested,[13] no work has been found with agents representing multiple variables or, in the case of MDO, having each discipline represented by an agent.

Probability Collectives are worthy of consideration as an automatic problem decomposition technique as it gives one method of ranking links between variables, or disciplines, in an implicit fashion. Strongly connected variables/agents would strongly affect each other's behavior as each seeks to maximize their own perceived reward for assuming a value. Methods may implicitly form modules by the semi-independent actions of agents, each representing either a piece of data or a discipline.[13]

Table 1 shows the reward obtained for each value selected for two input variables (agents). Each agent seeks to maximize their own observed reward by selecting a value that provides it. As each agent only controls one variable, each agent learns a one input response surface equation (RSE) to model only the averaged expected performance of the system due to its own actions. The agents for X_1 and X_2, in Table 1, model their received reward which is normally tied to the global problem. For this system, X_1 would be drawn to C and X_2 would observe a uniform distribution, Figure 7.

Table 1. Rewards Received by Two Example Agents, represented by X_1 and X_2.

	$X_1 = \text{A}$	$X_1 = \text{B}$	$X_1 = \text{C}$
$X_2 = \text{D}$	1	1	3
$X_2 = \text{E}$	3	0	2
$X_2 = \text{F}$	2	1	2
$X_2 = \text{G}$	0	4	1

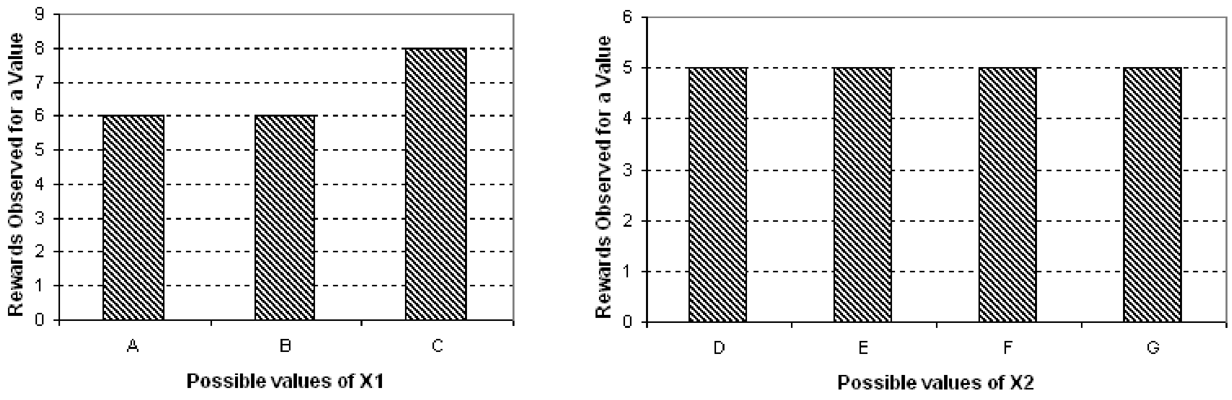


Figure 7. Two perspectives on the availability of rewards in the example space.

In Table 1, with the distribution for X_2 uniform from the perspective of the agent controlling variable X_2, the agent observes that any value for it is equivalent. The distribution for X_1 favors option C without added coupling information that might aid it in the discovery of $X_1 = \text{B}$ with $X_2 = \text{G}$. An annealing schedule, much like that used in simulated annealing, and sampling are typically used[13,14] to help avoid these local minima, but they could also be avoided by adding explicit coupling information.

Because the model for the solution and input space work with distributions it can be used with discrete or continuous problems. The method has also been used on constrained problems.[15] and on systems that are already inherently weakly coupled. It has been applied successfully to distributed control systems[13] and to aircraft flight routing.[16]

Currently, features in the solution space that deal with inter-agent coupling are only beginning to be examined. The method for optimizing the expected value from the modeled distributions may point the agents towards a local minimum. A more explicit treatment of interactions between variables in the solution space may help ameliorate the dependence of successful optimization on the selected annealing schedule.

II.C. Multi-Network Systems Applied to Problem Decomposition

II.C.1. Evolving Neural Networks for Problem Decompostion

Interesting recent work by Khare[17] in 2006 has sought to automatically decompose several problems by evolving neural networks (NN) for each sub-problem component. NNs 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 RSEs.[18] There are a large number of excellent introductions[19] to topic of neural networks (NN).

Regarding the expressive power of neural networks, there are proofs that show that there exists a network of two hidden layers that can approximate any arbitrary function.[20] A network with a single hidden layer and a sufficient (possibly very large) number of nodes can represent any continuous function and all boolean functions.[20] One would use a neural network when there are a large number of varied examples for the behavior one wants to learn. The set of data may contain errors as the training is robust to noise within the training set.[18] After a network is trained, the runtime evaluation of the regression is very rapid when compared to nearest neighbor type methods.[18]

The large training set and high number of weights cause long training times. The NN is also not able to explain its response to human examination. The network acts as a black box. Cases of medical diagnosis, for example, require an explained set of human-readable reasoning that is not available through a NN. There is also an extremely high probability that the training set will be overfit by the network during training. When a network overfits a set of data, both the information and noise from a set of data is fit by the model. Another group of data, called the validation set, is used to cut off the NN training once the network no longer improves its prediction of data not used during regression. 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.[19]

One method used for automatic problem decomposition is to seek to regress the behavior of subproblems automatically by using genetic operators. Each potential module is given a randomly chosen subset of the input variables and seeks to regress output behavior by the examples present in the networks training set. This attempts to decompose those subproblems that are solvable from input information alone; as opposed to those that require the output of other modules. All of these models will be competing with each other to form the building blocks that will be used by larger modular neural networks.[17] Here the combining of the modules is viewed as a subproblem in itself.[17]

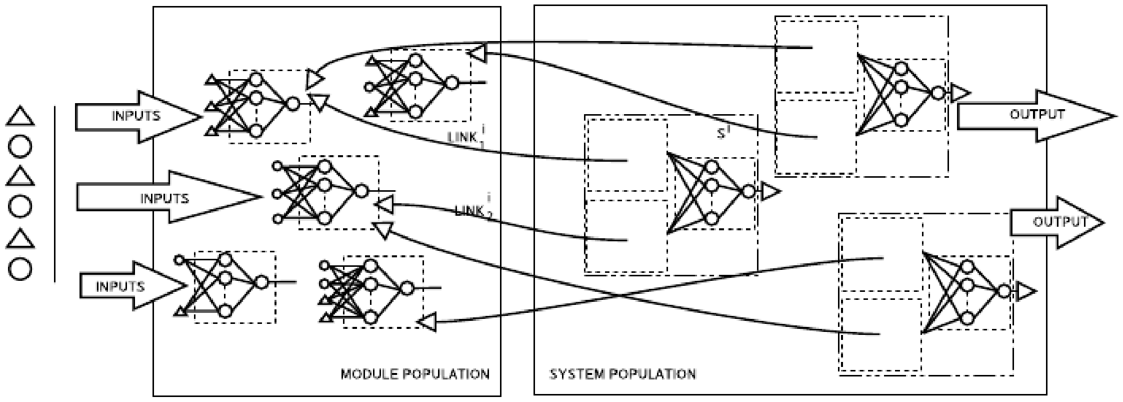


Figure 8. Two Populations Maintained to Generate Co-evolutionary Modules.[17]

As shown in Figure 8, the method used to select among these modules is to start with a general population of modules with each module accepting a random number of the input variables. Composite system models are created from these modules and each system model trains a combining network based off of training data, Figure 8.

The success of each system model, in predicting a validation set, provides a fitness measure for the model. The presence of modules in high performing system models provide a fitness measure for the module population. One could also have other fitness metrics for the module population such as a measure of how often a 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 two population model shown in Figure 8 does not scale well with problem size. One difficulty is due to the needs for a neural network structure. While neural networks have a great amount of representational power and are very quick, once trained; it can take a long time to initially regress their fit onto the behavior of their training data. This demand for training time means that the discovery of the correct number of subproblems does not scale well for this method.

In Khare[17] the maximum number of subproblems 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 subproblems automatically.[17] The genetically inspired searching method though was not able to scale well enough to set the upper limit at some very high number or to remove the upper limit from the work in the Thesis.[17] 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 valiables and with the automated discovery of subproblems.

II.D. Information Theoretic Decomposition in Engineering

II.D.1. A Minimum Discription Length Heuristic

Conceptually equivalent to the principle of Occam's razor;[21] the best hypothesis (fit of the data) is seen as the simplest able to describe a set of data 'well'. The measurement of 'simplicity' is approximated by the amount of information it would take to describe the hypothesis. Any regularity in the data could be used by a model to reduce the amount of data that must be retained to describe the data.[22] For instance, a data set containing one hundred values of the same integer could be greatly compressed into a model for that set. On the other hand, a message with a random sampling of numbers would require a larger model to describe the data. This heuristic seeks to balance between increasing the complexity of the model used and how 'well' the data is fit by the model.

II.E. Summary of Utility Heuristics Applied to Problem Decomposition

Currently, most methods have dealt with decomposing the problem before the running of an analysis by: treating all links as equally important[7] (binary connections), using heuristics such as ranking by the number of variables comprising the link,[9] and by using survey results from experts to rank linkage importance with a discrete value (ie. low/med/high).[5]

Global Sensitivity Equations have been used at runtime to define the total derivatives of the output responses in terms of the subsystem local sensitivities.[2] 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 allow the problem behavior to dictate the importance that should be assigned to interdependencies.

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 new methods would allow the field to tackle even larger problems.

Heuristics continue to be widely used to rank the importance of links, typically using a DSM representation.[1–5] The following is a representative list of the heuristics that have been used to automatically decompose problems in engineering.

Table 2: Overview of Published Utility Functions used for Problem Decomposition.

Decomp. Method	Problem Repres. and Optim. Method	Utility Metric Description	Utility Function	Reference
Implicit GA	Binary DSM with Basic GA	Minimize the number of feedback connections 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)[6]

Table 2: Overview of Published Utility Functions used for Problem Decomposition.

Decomp. Method	Problem Repres. and Optim. Method	Utility Metric Description	Utility Function	Reference
Implicit GA and Explicit Measurements	Weighted DSM with 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)[2]
Implicit GA and Explicit Measurements	Weighted DSM with Basic GA	The iteration factor is a user defined prediction as to how many times a feedback will be called. 2 iterations for very weak feedbacks; 8 for very strong feedbacks.	Ordering of tasks involve both GSE generated strengths and a user generated prediction as to how often an iteration linkage will be used. Given estimates for each analysis cost, attempts to find the least costly configuration.	Rogers [DeMAID-GA] (1996)[23]
Implicit GA	Discrete DSM with Basic GA	Links in matrix weighted by the number of variables passed within each linkage (thickness of pipe). Each variable seen as equally important. Also minimized the 'total length of feedback'.	Suggested the consideration of bandwidth concerns and database size. Objective used to compare against DeMAID for 'total length of feedback' was $f = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(j-i)w(i,j)$	Altus et al. [AGENDA] (1996)[9]
Implicit GA	Binary DSM with Basic GA	Summed distance of 1's from left of matrix (concurrency).	min $f = \sum_{i=1}^{n}\sum_{j=1}^{n}(j)v(i,j)$	Todd (1997)[24]
		Summed distance of 1's from bottom of matrix (concurrency).	min $f = \sum_{i=1}^{n}\sum_{j=1}^{n}(n-i)v(i,j)$	
		Summed distance of 1's above diagonal (reduce feedback).	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.	
Implicit GA	Discrete DSM with 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). n_1 is the index of the first component in the module. n_2 is the index of the last component in the module.	$MSI = MSI_i - MSI_e$ $MSI_i = \frac{\sum_{i=n_1}^{n_2}\sum_{j=n_1}^{n_2} w_{i,j}}{(n_2-n_1)^2-(n_2-n_1)}$ $MSI_e = \frac{\sum_{i=0}^{n_1}\sum_{j=n_1}^{n_2} w_{i,j}+w_{j,i}}{2n_1(n_2-n_1)} + \frac{\sum_{i=n_2}^{N}\sum_{j=n_1}^{n_2} w_{i,j}+w_{j,i}}{2(N-n_2)(n_2-n_1)}$	Whitfield et al. (2002)[4]
Implicit GA	Discrete DSM with Basic GA	Heuristics for tasks where part of one task can occur in parallel to a later task, often seen during scheduling tasks. 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 of themselves should.	Whitfield et al. (2005)[25]
Neural Network	Neural Networks with 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. Second population selects for a combining network.	Khare (2006)[17]
Information Theoretic	Discrete DSM with 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)[26]

III. Proposal to Adopt EDA Methods for Engineering Decomposition

Estimation of Distribution Algorithms (EDA) is an area of active research in computer science and could advance the state of problem decomposition in engineering by bringing new methods for the explicit computation of the importance of interactions. While the crossover operator in GA implements the classic idea of ‘divide and conquer’, the challenge of discovering where a crossover should occur is a stumbling block for the method. This has spurred the development of other methods such as PBIL[27] in 1995 and Pearl's foundational 1989 EDA paper[28] which would find likeminded researchers with MIMIC[29] in 1997, COMIT[30] in 1997 and BMDA[31] in 1999. Many of these methods use information theoretic modeling to estimate a distribution to generate inputs that will produce values from better performing areas of the solution space.[32]

What will be described here is the core approximation concept from Chow's 1968 work[33] utilized by Pearl's paper[28] for the structural modeling of problems. This idea also appeared later in Baluja's addition[34] to MIMIC, extending it with the use of Chow's dependency tree approximation of a distribution. These concepts will become clearer in the following section which will aim to provide a basis for programming an EDA.

III.A. Mutual Information Clustering

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. MIMIC is able to more efficiently converge over a design space by explicitly modeling the structural interactions between the input values. It uses prior solutions to build a model of the solution space that centers the peaks of a multimodal distribution over the areas that are likely to contain high performing candidate values. It builds this model distribution for the solution space by using statistics from prior samples from the true distribution over all solutions.

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

Equation 1 describes the exact joint distribution between a set of random variables. By approximating this function, information about the relationships between the variables can be exploited to search the domain space. The dependency tree version of MIMIC[34] uses pair-wise conditional probabilities to create an approximation to the true distribution above. 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).

This approximation could be made to use ternary-conditional probabilities for a more accurate match to the exact joint distribution but that would require far more data to accurately determine the ternary interactions.[29]

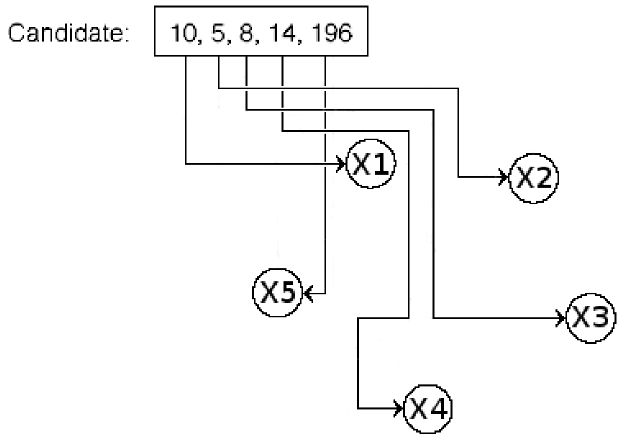


Figure 9. Specifying Nodes for Each input Variable.

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 serve as the approximated model, Figure 9. The mutual information between each pair of variables i and j is computed; this only requires knowledge of $p(x_i)$ and $p(x_i \ given \ x_j)$. The mutual information can be calculated from this information and used to weight the links between the variables. A graph with computed weights is shown in figure 10 for an example problem.

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.[35] The model with pair-wise statistics that

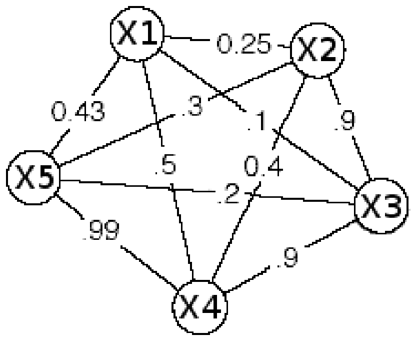


Figure 10. Fully Connected Graph Showing Mutual Information Between Each Node.

minimizes the KL distance between it and the true distribution will be the best possible approximation for the true distribution available.[33] 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[33] and Baluja.[34]

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, Figure 11. 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.

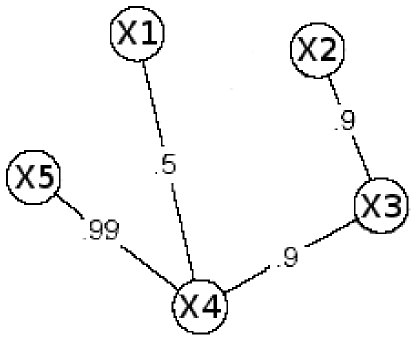


Figure 11. Maximum Spanning Tree for Mutual Information Between Each Node.

This tree model for the distribution is equivalent to the best approximated distribution to the true distribution, using only single and conditional probabilities. The tree shown in Figure 11 is equivalent to the distribution described by Equ. 2.

$$\hat{p}(X)_{approx} = p(X_4)p(X_5|X_4)p(X_1|X_4)p(X_3|X_4)p(X_2|X_3) \tag{2}$$

Sampling from the tree, a model of the solution space, is straight forward. 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 Equ. 2 comes from the tree shown in Figure 11; 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 becomes biased towards the higher performing areas of the solution space by using the lowest member of the top N percent of the data to specify the low-bar for data used to form the next generation's model. In MIMIC, N 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.

It is proposed that by explicitly modeling the interdependencies between input variables, such as in Figure 2, the weakly connected inputs can be found to aid in the decomposition of larger problems. This type of analysis could be performed with a DSM to evaluate the importance of intermediate variables to the outputs we are interested in. This would aid in the informed decomposition of DSMs.

The approach deals with inter-variable coupling in a principled manner rather than through the random separation of variables into sub-problems, such as in classical genetic algorithms. Recent advances to building trees based on sparse probability information could be transferred directly to speeding the overhead required to create this approximation for the solution space.[36] This would make the approach more useful for problems with cheaper fitness functions, purchasing this functionality with some added complexity during the methods implementation.

Though many of these EDA methods were originally designed to work with discrete values, nothing about the equations above require discrete probability distributions. Continuous values could be handled by adding an estimator to create a continuous probability function based on a discrete set of data. Constraints could be handled by disallowing samples that violate a constraint from being used in the model generation or by using penalty functions. This method for handling constraints may make reaching a solution resting against a constraint more difficult as the error between the true and modeled distribution would be larger in this area.

IV. Example Decomposition of a Coupled Analytic Problem

The four peaks problem was used by Baluja to describe the behavior of PBIL[27] and later by DeBonet for MIMIC.[29] 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 other value, see Figure 12. Leading and trailing values are explained by example in Equation 4. The worth of one variable in this coupled example is directly related to the values of many other binary input variables.

$$\bar{f}(\bar{X},T) = \begin{bmatrix} max[tail(0,\bar{X}), head(1,\bar{X})] + reward(\bar{X},T) \\ max[tail(0,\bar{X}), head(1,\bar{X})] + reward(\bar{X},T) + B \end{bmatrix} \tag{3a}$$

$$tail(0,\bar{X}) \text{ equals the number of trailing 0s in } \bar{X} \tag{3b}$$

$$head(1,\bar{X}) \text{ equals the number of leading 1s in } \bar{X} \tag{3c}$$

$$reward(\bar{X},T) = \begin{cases} N \text{ if } tail(0,\bar{X}) > T \text{ and } head(1,\bar{X}) > T \\ 0 \text{ otherwise} \end{cases} \tag{3d}$$

$$Example\ Candidate: \begin{bmatrix} \underbrace{11111}_{\text{head=5}}\, 01010111\, \underbrace{000000}_{\text{tail=6}} \end{bmatrix} \tag{4}$$

This example problem allows the toggling of the problem size, by changing the size of X, 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 X. The solution space formed by this is shown in Figure 12 for a 100 input vector X. As the value of each input depends on the values of several other inputs, methods that explicitly model the interdependencies between inputs should be able to leverage this information to converge with fewer function calls.

A mature third-party implementation of genetic algorithms[37] 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.

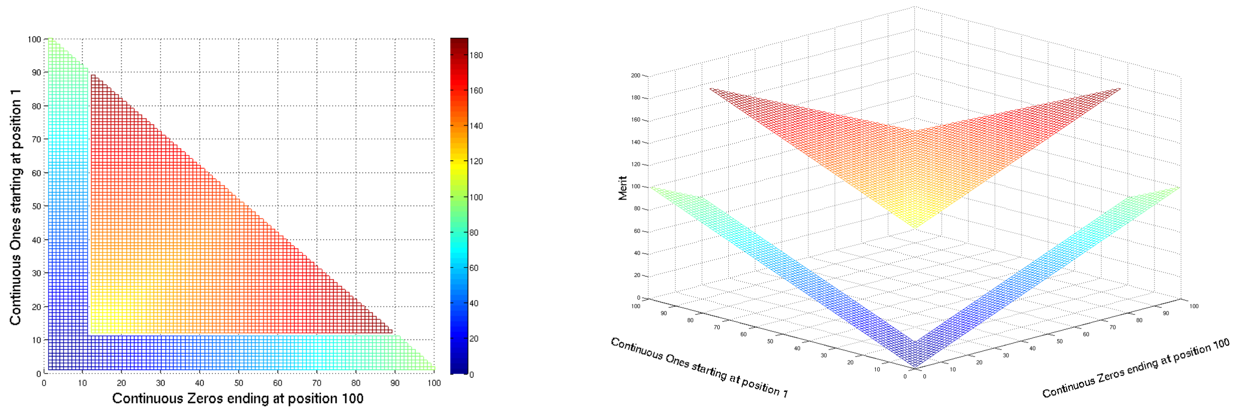


Figure 12. Four Peaks solution space for an input X of size 100, T=10.

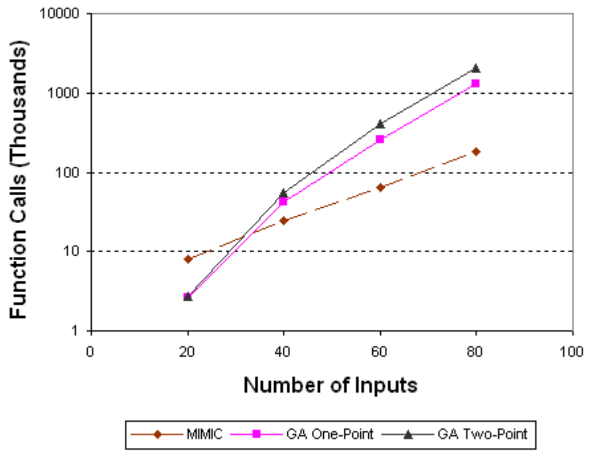


Figure 13. Comparison of the Number of Function Calls Required to Find the Global Optimum.

Genetic algorithms are often used in engineering for multimodal domains and to apply decomposition heuristics for coupled problems. As covered earlier, the pair-wise interactions between variables are implicitly modeled by the crossover operation. The crossover operation was created for the carrying forward of sub-problem solutions but Figure 13 shows the challenge two crossover operators (single and two-point crossover) have on the Four Peaks Problem. Explicitly modeling the pair-wise interdependencies between variables allows MIMIC to obtain an order of magnitude improvement when working with 80 inputs, Figure 14.

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 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 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’ correctly placed at the first variable, to potentially separate the two components of this problem by 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 allow for solutions at a tenth the cost of the two-point crossover. Even without the application of mutual information clustering to the ranking of links in a DSM, so many heuristics use GAs that MIMIC and other EDA methods 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, One-Point, and 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 values for each link could be stored and related to the solution domain in a similar manner. Links could be ranked by the information they provide toward solving the objective function. Pair-wise information between links would provide a new layer of information for developing new decomposition techniques.

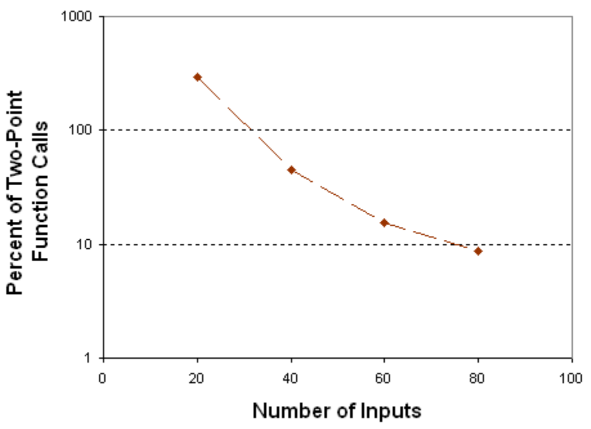


Figure 14. Function Calls Required by MIMIC, as a Percentage of Two-Point Function Calls.

One strong source of flexibility for EDAs 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 EDAs 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. The KL distance between domain distributions could also be used as a metric for how closely the developed approximate model matched the actual problem behavior.

V. Conclusion

This work has aimed toward better educating engineers in the current challenges of problem decomposition within engineering, with an examination of the previous work done within the field. It has also proposed a promising new avenue of research, using EDA techniques to decompose highly coupled engineering problems. It is hoped that by better exposing this topic, this paper will spur greater collaboration on these challenges. Although GAs have been successful in many domains, the technique can experience challenges with coupled multi-modal problems. As has been shown, the addition of pair-wise statistics allows for a principled global method that is better able to scale, over the current state of the practice, on large coupled problems.

References

[1]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.

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

[3]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.

[4]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.

[5]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.

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

[7]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.

[8]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.

[9]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.

[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]Srinivas, M. and Patnaik, L. M., "Genetic Algorithms: A Survey," *Computer*, Vol. 27, No. 6, June 1994, pp. 17–26.

[12]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.

[13]Bieniawski, S. R., *Distributed Optimization and Fight Control using Collectives*, Ph.D. thesis, Stanford University, October 2005.

[14]Lee, C. F. and Wolpert, D. H., "Product Distribution Theory for Control of Multi-agent Systems," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, New York, July 2004, pp. 522–529.

[15]Bieniawski, S. R., Kroo, I. M., and Wolpert, D. H., "Discrete, Continuous, and Constrained Optimization Using Collectives," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, August 2004.

[16]Antoine, N. E., Bieniawski, S., Wolpert, D. H., and Kroo, I., "Fleet Assignement Using Collective Intelligence," *42nd AIAA Aerospace Sciences Meeting*, Reno, NV, January 2004, AIAA Paper 2004-0622.

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

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

[19]Mitchell, T. M., *Machine Learning*, The McGraw-Hill Companies, Inc. and MIT Press, 1997.

[20]Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, December 1989, pp. 303–314.

[21]Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K., "Occam's Razor," *Information Processing Letters*, Vol. 27, April 1987, pp. 377–380.

[22]Yu, T.-L., *A Matrix Approach for Finding Extrema: Problems with Modularity, Hierarchy, and Overlap*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2006.

[23]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.

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

[25]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.

[26]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.

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

[28]Pearl, J. and Dechter, R., "Learning Structure from Data: A Survey," *Preceedings COLT '89*, Santa Cruz, Ca., August 1989, pp. 230–244, Technical Report R-132.

[29]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.

[30]Baluja, S. and Davies, S., "Fast Probabilistic Modeling for Combinatorial Optimization," *Proceedings of the National Conference on Artificial Intelligence*, Vol. 15, No. 15, 1998, pp. 469–476.

[31]Pelikan, M. and Muehlenbein, H., "The Bivariate Marginal Distribution Algorithm," *Advances in Soft Computing: Engineering Design and Manufacturing*, edited by R. Roy, Springer, 1999, pp. 521–535.

[32]Larranga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms. A new Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2001.

[33]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.

[34]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.

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

[36]Meilă, M., "An Accelerated Chow and Liu Algorithm: Fitting Tree Distributions to High Dimensional Sparse Data," *Proc. 16th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, San Francisco, CA, 1999, pp. 249–257.

[37]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/.