

DETC2003/DTM-48657

A GENETIC ALGORITHM FOR DEVELOPING MODULAR PRODUCT ARCHITECTURES

Tian-Li Yu⁺, Ali A. Yassine, David E. Goldberg

University of Illinois at Urbana-Champaign
Department of General Engineering, Urbana, IL 61801

ABSTRACT

The architecture of a product is determined by both the elements that compose the product and the way in which they interact with each other. In this paper, we use the design structure matrix (DSM) as a tool to capture this architecture.

Designing modular products can result in many benefits to both consumers and manufacturers. The development of modular products requires the identification of highly interactive groups of elements and arranging (i.e. clustering) them into modules. However, no rigorous DSM clustering technique can be found in product development literature.

This paper presets a review of the basic DSM building blocks used in the identification of product modules. The DSM representation and building blocks are used to develop a new DSM clustering tool based on a genetic algorithm (GA) and the minimum description length (MDL) principle. The new tool is capable of partitioning the product architecture into an "optimal" set of modules or sub-systems. We demonstrate this new clustering method using an example of a complex product architecture for an industrial gas turbine.

(Keywords: Design Structure Matrix (DSM), Product Architecture, Modular, Integral, Genetic Algorithm (GA), Minimum Description Length (MDL)).

1. INTRODUCTION

Product architecture is defined as the scheme by which decomposed elements of a product are arranged in modules [1]. The less interacting these modules are, the more modular the product architecture is. Modularity as a strategy can be

employed for different reasons. In the right circumstances it allows increased product or organizational variety [2], increased rates of technological innovation [3], increased opportunities for market dominance through interface capture, and increased specialization at firm level which in turn may allow more flexible response to environmental change.

The development of modular products requires the identification of highly interactive groups of elements and clustering them into product modules. Clustering techniques are scarce in product development literature. However, the few ones found share two major deficiencies. First, the algorithms are manual, very dependent on human expertise, and consequently hard to automate or replicate [4, 5]. Another significant problem deals with the use of simple mathematical constructs to discriminate between product modules and consequently these algorithms collapse when confronted with complex product architectures [6, 7]. The goal of this paper is to develop a new DSM clustering technique that can be easily automated, yet sophisticated enough to handle complex architectures.

This paper uses the design structure matrix (DSM) to represent product architectures. Then, this representation is used to develop modular product architectures by clustering the DSM. The proposed clustering method is developed based on a genetic algorithm (GA) [8] and the minimum description length (MDL) principle [9, 10]. The algorithm is capable of partitioning the product architecture into an "optimal" set of modules or subsystems and can be fine tuned to mimic clustering arrangements proposed by human experts performing manual clustering of the DSM.

⁺ Corresponding author: Illinois Genetic Algorithm lab,
University of Illinois at Urbana-Champaign,
Tel. (217) 3332346, email: tianliyu@illgal.ge.uiuc.edu

The rest of the paper proceeds as follows. The next section defines the notion of modularity as used in this paper. Then, we provide a quick overview of the DSM method and the basic building blocks used for the identification of product modules, in Sections 3 and 4, respectively. Section 5 discusses some of the clustering difficulties encountered when using DSMs. In Section 6, we propose an objective function to evaluate different architectural arrangements based on the MDL principle. Section 7 introduces the genetic algorithm used to search for optimal product architectures. In Section 8, we demonstrate this new clustering method using an example of complex product architecture, and finally, Section 9 concludes the paper.

2. PRODUCT MODULARITY

Modularity is an ambiguous and elusive notion that has been loosely used in different ways by different people at different times. In its architectural usage, it is easier to define as an antonym, i.e. modular is the opposite of integrated. Webster's New World Dictionary defines the noun "module" as "*a compact assembly functioning as a component of a larger unit*" and the adjective "modular" as "of a module". Alternatively, the adjective "modular" is defined as "*whole or complete*". So at the two extremes, modular architecture is one made up of assemblies of components, whilst an integrated architecture is one made up purely of the lowest level of component without having intermediate assemblies. However, we emphasize that there is a continuum between a fully modular and a fully integrated architecture.

A fully modular architecture is one with clear clusters of elements, and where the relationships between the elements within an assembly are hidden to the elements outside the assembly. This incorporates the notion that a module not only contains elements, but also contains a higher density of relationships between those elements than to elements outside the module. The significance of being a modular cluster is that all the parts within the module possess the same relationships with each other and with parts outside of the module – this can be thought of as being the design rules of the module.

3. THE DESIGN STRUCTURE MATRIX (DSM) METHOD

A DSM is a matrix representation of a directed graph that represents a complex system. The nodes of the graph correspond to the column and row headings in the matrix. The arrows correspond to the "X" marks inside the matrix.¹ For example, if there is an arrow from node C to node A, then a 'X' mark is placed in row A and column C. Diagonal elements have no significance and are normally blacked-out or used to store some element-specific attribute(s). Alternatively, number "one"

can be placed instead of an "X" and "zero" instead of a blank. This makes the DSM a binary matrix with entries $d_{ij} = 0$ or 1.

Once the DSM for a product is constructed, it can be analyzed for the identification of modules; a process referred to as clustering. The goal of DSM clustering is to find subsets of DSM elements (i.e., clusters or modules) that are mutually exclusive or minimally interacting. In other words, clusters contain most, if not all, of the interactions (i.e., DSM marks) internally and the interactions or links between separate clusters is eliminated or minimized [6]. In which case, the blocks become analogous to team formations or independent modules of a system (i.e., product architecture). As an example, consider the DSM in Figure 1. The entries in the matrix represent the frequency and/or intensity of communication exchanged between the different development participants, represented by person A, person B, ...etc. As can be seen in Figure 1(b), the original DSM was rearranged to contain most of the interactions within two separate blocks: AF and EDBC. However, three interactions are still outside any block (i.e., team or module) and constitute the points of interaction/collaboration between the two teams (or interface between the two product modules). An alternative arrangement is suggested in Figure 1(c). This arrangement suggests the forming of two overlapping modules (i.e., AFE and EDBC).

Early clustering algorithms and underlying principles are described in [11]. These algorithms work in either a top down or a bottom up manner, either by cleaving along the line of least resistance or by aggregating along the path of greatest return. In either case, the relevant metric is some function of the number of linkages that cross a partition boundary. In Alexander's work, the objective function was implemented with a simple hill climbing search strategy whereby the system either was successively partitioned into ever-smaller sets or successively aggregated into ever larger sets, one element at a time. Further details of clustering algorithms and a wide range of applications can be found in [12].

Fernandez [6] used a similar approach to Alexander's bottom-up aggregation. Each element is placed in an individual set and bids evaluated from all the other sets (clusters). If any cluster is able to make a bid that is better than the current base case then the element is moved inside the cluster. The objective function is therefore a trade-off between the costs of being inside a cluster and the overall system benefit.

Sharman et al. [13, 14] attempted using the clustering algorithm described in Fernandez [6] on an industrial gas turbine. However, they showed that this algorithm falls short of being able to accurately predict the formation of "good" clustering arrangements for complex product architectures due to the oversimplification of the objective function utilized and the frequent susceptibility of the search algorithm used to be

¹ There are different ways of building a DSM. For a full description, please refer to the MIT DSM website at <http://web.mit.edu/dsm>.

trapped in local optimal solutions. This motivated the investigation of more complex objective function formulations and the utilization of more robust search strategies as no rigorous DSM clustering tools seems to exist in the literature.

	A	B	C	D	E	F	G
A	X	X			X	X	
B		X		X			X
C		X	X	X			X
D		X	X	X	X		X
E				X	X	X	
F	X				X	X	
G		X	X	X			X

(a) Original DSM

	A	F	E	D	B	C	G
A	X	X					
F	X	X					
E		X	X				
D			X	X	X	X	
B				X	X		X
C				X	X	X	
G				X	X	X	

(b) Clustered DSM

	A	F	E	D	B	C	G
A	X	X					
F	X	X					
E		X	X				
D			X	X	X	X	
B				X	X		X
C				X	X	X	
G				X	X	X	

(c) Alternative Clustering

Figure 1: Clustering examples

4. DSM BUILDING BLOCKS: PRODUCT ARCHITECTURE TERMINOLOGY

In this section, we review the basic syntax and semantics for analyzing DSMs in order to characterize the architecture of a complex product and identify its modules. The analysis proceeds using simplified DSMs and product architectures. These simple constructs will be used as building blocks for analyzing more complex architectures.

Consider the simple product architecture depicted by the physical schematic of Figure 2(a). It shows five related (i.e.

connected) parts; four parts are connected to a fifth.² The situation may be visualized as E being some form of system level integrating component, or bus, which is connected to parts A, B, C, and D. In this instance, the relationships between the parts are symmetrical, i.e. part A depends on part E in the same way that part E does on A. To the right of the schematic the DSM for this system is drawn. To the right of the DSM a more conceptual architectural diagram that represents the same situation is shown. This third diagram can be drawn in any orientation as the relationships are symmetrical, thus at this stage no value should be assigned to any convention that chooses to draw part E above rather than below or to the side.

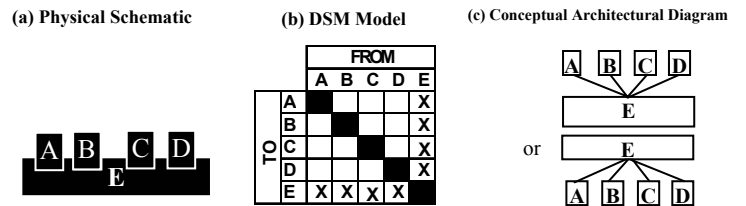


Figure 2: Simple bus, four chunks, and no modules

A part is the smallest possible decomposition of something whilst a chunk or element could be assemblies in their own right. For this reason strict terminology should differentiate between primitive elements (i.e. parts) or higher-level elements. We term a bus as being ‘simple’ when it is a primitive element. Figure 3 introduces more relationships between the parts. In this instance, the pair of parts A and B is related symmetrically to each other as well as to part E. A similar structure can be seen in the pair of parts C and D. In the DSM, this pairing is defined as being a modular cluster and denoted by “Module SS” and “Module TT”.³

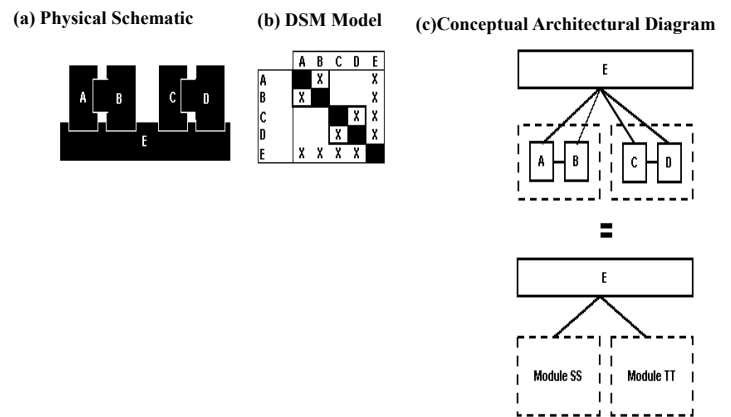


Figure 3: Simple bus, four chunks, and two modules

² The relationship or connectivity can be thought of as representing a flow of mass, energy, information, or force/geometrical constraint between the parts.

³ Note that buses may also be comprised of multiple primitive elements, in which case they may be referred to as a ‘bus module’.

This example also illustrates the notions of “pinning” and “holding away.” Here B is pinned in place between A and C by its relationship with A and C. This is a common and easily appreciated situation in much physical architecture. A less easily appreciated situation is that of “holding away,” which is seen here in that element A is held away from element C by element B. The combination of these two notions assists in describing B. The combination of these two notions assists in describing the drawbacks of various clustering arrangements and clustering heuristics. Pinning can only occur to compound elements (modules) that have relationships with two modules, whilst any primitive element can be held away.

Many architectures are more complicated than the examples inspected above. Figure 4 shows a number of possibilities. Either (A, B) and (C, D) can be concatenated into modules, the sequence A through F can be thought of as one super module, an intermediate module TT can be sandwiched between module SS (comprising A, B) and module UU (comprising E, F), or simply described as being comprised of the primitives A through F with the bus module (comprising G, H).

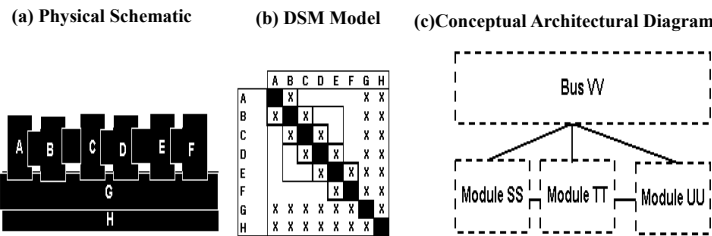


Figure 4: Imperfection, pinning, and holding away

5. DSM CLUSTERING COMPLEXITIES

The problem with applying automated clustering algorithms to more complex DSMs is that they find it hard to extract the relevant information from the data, and then to convey it to the user. This is most noticeable in the poor handling of pinned modules, buses, and path-dependent situations. We investigate this further in the rest of this section.

5.1 Path Dependency in Clustering

Consider a triangular arrangement of symmetrical relationships that can be loosely clustered into three similar modules AA, BB, and CC, as shown in Figure 5. The DSM for this physical arrangement can only show two of the real clusters and must break up the third. The way in which the clustering algorithm operates will be path dependent inasmuch as once it has started to cluster on any two nodes it is unlikely to reverse into a different configuration. This situation may occur because of the way in which raw data is presented to the clustering algorithm (for example branch and bound algorithms that are presented with a partially clustered starting point may never branch widely enough to evaluate alternative solutions) or may arise through chance if a perfectly random starting point is presented

to an algorithm that makes an initial random guess. In the example of Figure 5, cluster CC has been broken up even though it is identical in all respects to the other two clusters. It could as easily have been AA or BB clusters that were broken up by being positioned where CC is.

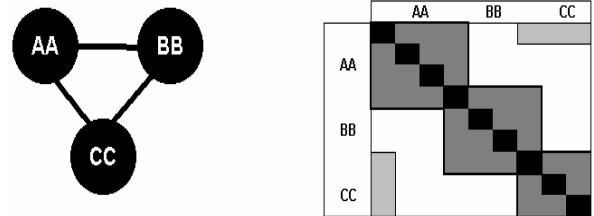


Figure 5: Planar triangular clusters

5.2 Dimensions & Topology

Consider a simple three-dimensional structure such as a tetrahedron or three-dimensional pyramid. This is depicted in Figure 6 showing four equal clusters, each with dense internal relationships and weaker (or sparser) external relationships. As before, if all the clusters are perfectly equal it is purely a matter of chance how any clustering algorithm would present an answer. In this example, cluster DD is the one that is visually disrupted most by being presented last in the sequence. This has the effect of spreading its inter-cluster relationships over a wider spatial area, which is depicted in the macro-scale DSM as being lower density blocks of grey. To an untrained observer this might be thought to be a bus structure where cluster DD is the unique possessor of system wide integrating functions and some semi-random cross-linking occurs in the zone AA-CC.

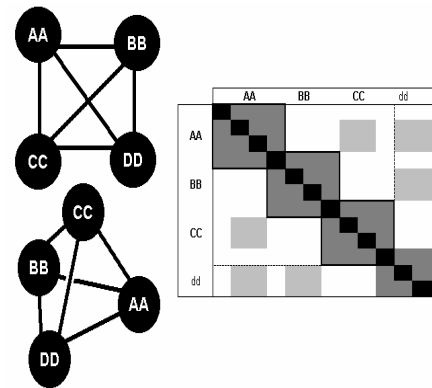


Figure 6: Tetrahedron of clusters

6. MINIMUM DESCRIPTION LENGTH (MDL) BASED OBJECTIVE FUNCTION

As mentioned earlier, previous DSM clustering algorithms can be found in [6, 13]. Their results showed that the objective function used was short of accurately predicting “good” clustering because of the oversimplified objective function and search strategy used. The lack of an efficient DSM clustering method, evidenced from prior DSM literature, motivated us to

seek better criteria to guide DSM clustering. Based on the previous two sections, we develop the following criteria for the development of a new clustering algorithm:

1. The algorithm should be able to suggest the optimal number of clusters.
2. The algorithm should be able to detect the existence of bus modules.
3. The algorithm should be able to detect overlapping clusters and three dimensional structures.

While the above two requirements can be addressed by the appropriate choice of a clustering objective function, as will be discussed next, the third requirement is directly related to the proposed genetic algorithm, discussed in Section 7.

6.1 Model Description

Suppose we have a model which describes a given data set, $DSM=[d_{ij}]$. Here, the model means a description that specifies which node belongs to which cluster and/or a bus. Usually, the model does not describe completely the given data; otherwise, the model is too complicated. Therefore, the description length that the model needs to describe the given data consists of two parts: the *model description* and the *mismatched data description*. This scheme may be easier to understand in light of the following sender-receiver example.

Assume a sender has a given data set which is needed by the receiver. Given a model that approximately (i.e. not exactly) describes the given data set, the sender first sends the model (i.e., model description) to the receiver. To ensure that the receiver gets exactly the same data set, the sender is also required to send the data which is mis-described (i.e., mismatched data description) by the model sent earlier. If the model is too simple, the model description is short; but many data mismatches exist and the mismatched data description becomes longer. On the other hand, a complicated model reduces mismatched data, but the model description is longer.

The minimum description length (MDL) principle [9] satisfies our needs for dealing with the above dilemma. The MDL can be interpreted as follows: *among all possible models, choose the model that uses the minimal length for describing a given data set (that is, model description length plus mismatched data description length).*

There are two key points that should be noted when MDL is used: (1) the encoding should be uniquely decodable, and (2) the length of encoding should reflect the complexity. For example, the encoding of a complicated model should be longer than that of a simple model. Next, we define the MDL clustering metric in detail.

6.2 Model Encoding

The way we encode the model is naïve. The description of each cluster starts with a number which is sequentially assigned to each cluster, and then this is followed by a sequence of nodes in the cluster. Figure 7 shows a DSM clustering arrangement and the corresponding model description. It is easily seen that the length of this model description is as follows:

$$\sum_{i=1}^{n_c} (\log n_c + \log n_n + c_{l_i} \cdot \log n_n),$$

where n_c is the number of clusters in the DSM, n_n is the number of nodes, c_{l_i} is the number of nodes in the i^{th} cluster, and the logarithm base is 2. In the example of Figure 7, $n_c = 2$ clusters, $n_n = 8$ nodes, $c_{l_1} = 3$, and $c_{l_2} = 4$. The table in the figure reads as follows: “cluster 1 has 3 nodes: B, D, and G; cluster 2 has 4 nodes: A, C, E, and H”.



Length	$\log n_c$	$\log n_n$	$3 \log n_n$	$\log n_c$	$\log n_n$	$4 \log n_n$
Description	1	3	B,D,G	2	4	A,C,E,H

Figure 7: Above is a clustering arrangement of a DSM. Below is the associated model description

If n_n and n_c are known, it is not difficult to see that the above model description is uniquely decodable. n_n is given, and by assuming $n_c \leq n_n$, $\log n_n$ bits are needed to describe n_c . The $\log n_n$ bits are fixed for all models, and therefore they are omitted without loss of accuracy.

6.3 Mismatched Data Description

Based on the model, we first construct another DSM (call it DSM'), where each entry d'_{ij} is “1” if and only if:

1. some cluster contains both node i and node j simultaneously, or
2. The bus contains either node i or node j .

Then, we compare d'_{ij} with the given d_{ij} . For every mismatched entry, where $d'_{ij} \neq d_{ij}$, we need a description to indicate where the mismatch occurred (i and j) and one additional bit to indicate whether the mismatch is zero-to-one or one-to-zero. Let us define the two mismatch sets

$S_1 = \{(i, j) \mid d_{ij} = 0, d_{ij}' = 1\}$ and $S_2 = \{(i, j) \mid d_{ij} = 1, d_{ij}' = 0\}$. We call the mismatch that contributes to S_1 the *type 1 mismatch*, and the mismatch that contributes to S_2 the *type 2 mismatch*. The mismatched data description length is given by:

$$\sum_{(i,j) \in S_1} (\log n_n + \log n_n + 1) + \sum_{(i,j) \in S_2} (\log n_n + \log n_n + 1).$$

The first $\log n_n$ in the bracket indicates i , the second one indicates j , and the additional one bit indicates the type of mismatch.

6.4 MDL Clustering Metric

The MDL clustering metric is given by the weighted summation of the model description length given in Section 6.1 and the mismatched data description given in Section 6.2. With some arithmetic manipulations, the metric can be written as follows:

$$f_{DSM}(M) = (1 - \alpha - \beta) \cdot \left(n_c \log n_c + n_c \log n_n + \log n_n \sum_{i=1}^{n_c} c l_i \right) + \alpha \cdot [|S_1| (2 \log n_n + 1)] + \beta \cdot [|S_2| (2 \log n_n + 1)],$$

where α and β are weights between 0 and 1.⁴ The α and β setting is not a simple task. A naïve setting is $\alpha = \beta = 1/3$. In Section 8, we adjust α and β to mimic the behavior of a manual clustering arrangement.

Finally, the objective is to find a model M that minimizes f_{DSM} . In other words, f_{DSM} is the length (in bits, when the logarithm is taken in base 2) that model M needs to describe the given data.

7. SEARCH STRATEGY – GENETIC ALGORITHMS

Exhaustive search is the most basic search method. It generates all possible architectures for a given product and exhaustively evaluates each one to determine the best one. This search method is only possible for small DSM sizes, as the number of possible product architectures grows very large, very fast, as the DSM size increases.⁵ Even with today's high-powered computers, using an exhaustive search to find the optimal solution for even relatively small problems can be prohibitively expensive.

⁴ Here we use weighting in order to match the preference of human experts. This becomes more evidential in the case study.

⁵ The number of possible product architectures for a system with n elements

without cluster overlap is: $\sum_{i=1}^n C_i^n = 2^n - 1$.

Genetic algorithms are general-purpose search algorithms based upon the principles of evolution observed in nature [8]. Genetic algorithms are able to find optimal or near optimal solutions by using natural mechanisms, such as selection, crossover, and mutation. This section gives an introduction to the particular simple genetic algorithm used in this paper. Encoding and some illustrative examples are also given.

It is worth noting that genetic algorithms have been used on DSMs by other researchers [15, 16]. However, these algorithms solve a different type of DSM related problem called DSM partitioning or sequencing. Partitioning refer to transforming a DSM into a block triangular form.

7.1 Introduction to a Specialized Genetic Algorithm

In this paper, we adopted a genetic algorithm with *uniform crossover*, *mutation*, and $(\lambda + \mu)$ *selection*. Below is an outline of how the specialized genetic algorithm uses these operators to search for optimal solutions:

1. Create an initial population of chromosomes. Each chromosome is made up of a collection of genes (the parameters to be optimized) and represents a complete solution to the problem at hand. The gene values are usually initialized to random values within user-specified boundaries. Initially, there are λ chromosomes.

2. Crossover chromosomes to produce new offspring chromosomes. This crossover occurs according to some user-defined probability p_c (usually high) and results in new chromosomes having characteristics taken from both of the parent chromosomes. In this paper, *uniform crossover* is adopted. Uniform crossover operator randomly switches each gene of the two parent chromosomes with a certain probability (0.5, in this paper) to produce two new offspring. If an offspring takes the best parts from each of its parents, the result will likely be a better solution. The two parents are randomly picked up without (replacement) from the λ chromosomes, and the reproduction is continued until μ offspring chromosomes are produced. Note that no selection is performed here yet.

3. Mutate the genes of the offspring chromosomes. This mutation occurs according to some user-defined probability p_m (usually low) and serves to introduce some variability into the gene pool. For a binary encoding chromosome, mutation inverts the value of genes (from 0 to 1, or from 1 to 0) with the mutation probability p_m . Without mutation, offspring chromosomes would be limited to only the genes available within the initial population.

4. Evaluate each of the chromosomes in both parent chromosomes and offspring chromosomes. Each chromosome is evaluated by a fitness function (i.e., objective function as described in Section 6) to determine the quality of the solution. Note that only μ chromosomes need to be evaluated except for the first generation.

5. $(\lambda + \mu)$ selection is performed to select chromosomes that will have their information passed on to the next generation. In step 5, totally $(\lambda + \mu)$ chromosomes are evaluated. $(\lambda + \mu)$ selection chooses λ best chromosomes from the $(\lambda + \mu)$ chromosomes and passes them to the next generation. Note that elitism is embedded in $(\lambda + \mu)$ selection.

Repeat steps 2 through 5 until some termination condition has been met. This termination condition can be based simply on the number of generations or it can be based upon more complex criteria such as the fitness convergence.

7.2 Illustrative Examples

7.2.1 Encoding and Parameters Settings

As indicated in Section 6, the encoding should deal with overlapping clusters and three dimensional structures. As long as the encoding allows that a node belongs to several different clusters, the GA is able to detect overlapping clusters and three dimensional structures.

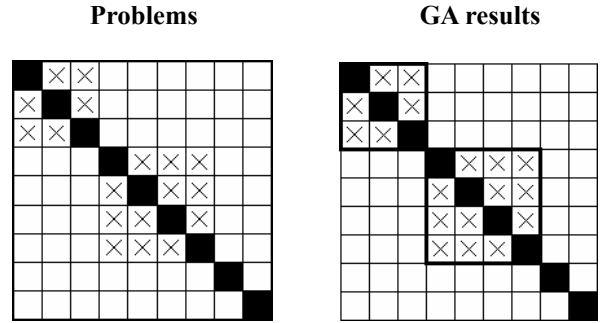
The chromosome is a binary string of $(c \cdot n_n)$ bits, where c is predefined maximal number of clusters, and n_n is the number of nodes. The $(x + n_n \cdot y)$ -th bit represents that node $(x+1)$ belongs to cluster $(y+1)$. The last cluster is treated as a bus. For example, in Figure 7: $n_n=8$, and given c is 3, then the model might be described by the following chromosome shown in Figure 8. When manipulated, the chromosome is transformed into a binary string which is a concatenation of all rows.

Node	A	B	C	D	E	F	G	H
Cluster 1	0	1	0	1	0	0	1	0
Cluster 2	1	0	1	0	1	0	0	1
Bus	0	0	0	0	0	0	0	0

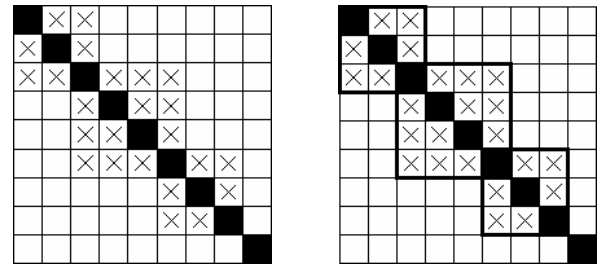
Figure 8: A chromosome that represents the model shown in the lower part of Figure 7. The binary string manipulated is 010100101010100100000000.

7.2.2 GA Results of Illustrative Examples

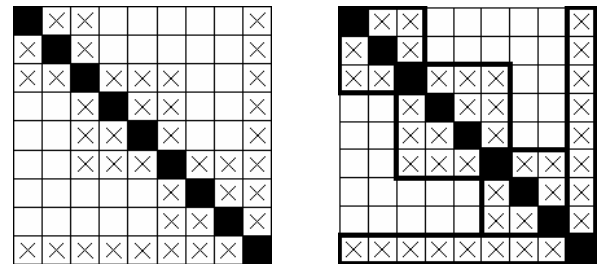
In the following experiments, the number of nodes is 9, the maximal number of clusters is set to 4, and hence the chromosome length is $4 \times 9 = 36$. The crossover probability p_c is 1, mutation probability p_m is $1/36$, and a $(10+100)$ selection is adopted. The GA is terminated if no improvement happens for five generations. The weights in the MDL clustering metric are set equally to $1/3$. Figure 9 shows the GA results of some illustrative examples.



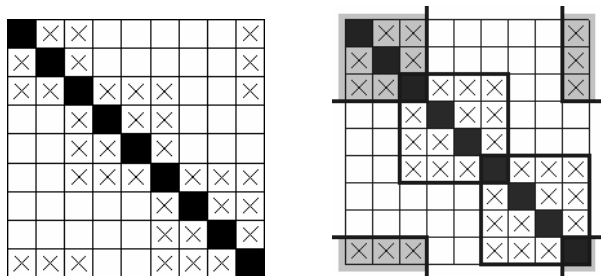
(a) Two non-overlapping clusters.



(b) Three overlapping clusters.



(c) Three overlapping clusters with a bus.



(d) Three dimensional overlapping clusters. The shadowed part forms a cluster

Figure 9: Illustrative examples. The left column is the given DSM, and the right column shows how the GA clusters it.

Figure 9(a) is the simplest case, two non-overlapping clusters. Figure 9(b) demonstrates the ability of identifying overlapping clusters. In Figure 9(c), a bus is introduced, and the genetic algorithm is able to identify it. Finally, the DSM in Figure 9(d) resembles the DSM in Figure 9(c); however, the result is totally different. The genetic algorithm recognized it as three overlapping clusters instead of a bus. Figure 9(d) also

demonstrates the ability of identifying three dimensional structures. It is interesting to note that the DSMs given in Figures 9(c) and 9(d) are similar but the clustering results are quite different. These results show that the GA with the MDL-metric is able to solve complex problems with overlapping clusters, a bus, or three dimensional structures.

8. REAL-WORLD CASE STUDY

A DSM for a generic 10 MWe gas turbine driven electrical generator set was constructed by decomposing it into 31 sub-systems. The sub-systems initially were listed randomly in the DSM and then tick marks denoting material relationship from one sub-system to another were inserted [17].

8.1 Manual Clustering of DSM

Intuitive manual clustering can yield different results depending on the extent to which a single group of system-wide relationships is emphasized over “good” clusters. One alternative arrangement is shown in Figure 10. The manual generation of this arrangement is discussed next.

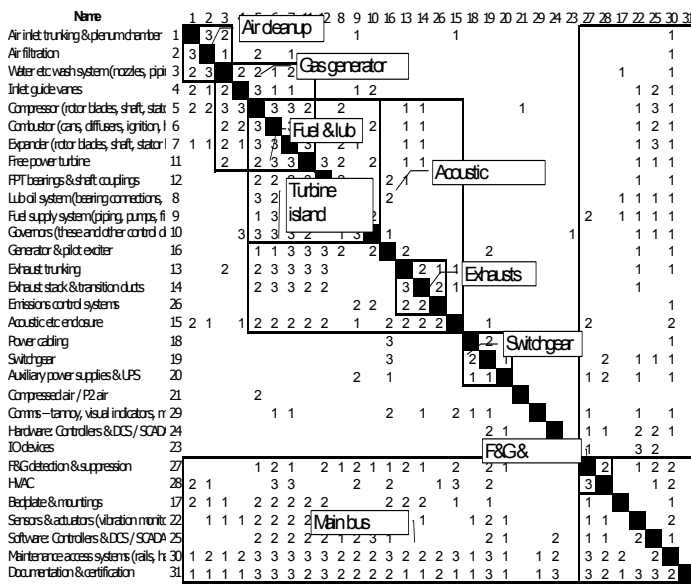


Figure 10: Physical DSM and manual clustering of 10MWe gas turbine with named clusters⁶

Figure 10 shows a previously reposted attempt at manually clustering the DSM [13, 14]. This took few manual changes to the order of elements in the initial DSM, which revealed the clusters marked in the figure. After inspection of the clusters, they were given names to identify them. Notice how some clusters are isolated (e.g. the switchgear) whilst others overlap (e.g. air clean-up with the gas generator) or are completely

⁶ The numbers (1, 2, and 3) in the figure represent varying dependency strengths. However, in this paper, we only consider binary DSMs. As such, our clustering algorithm for this DSM will treat all the dependency marks as equal.

embedded in a larger cluster (e.g. turbine island within the acoustic sources).

8.2 Automated Clustering of DSM Using the Proposed GA

In Figure 10, if we consider all entries in clusters and the bus should be 1 and all entries outside should be 0, then we can obtain the preference of human clustering: $|S_1|=190$ and $|S_2|=35$, where S_1 and S_2 are defined in Section 6.2. In other words, human tends to endure the type 1 mismatch more than the type 2 mismatch. According to the observation, we set $\alpha:\beta=35:190$ in the MDL clustering metric. By keeping $(1-\alpha-\beta)=1/3$ (the weight of the model description length remains the same), we can obtain $\alpha \approx 0.1037$ and $\beta \approx 0.5630$. The maximal number of clusters is set to 6, and we have 31 nodes which yields a 186-bit chromosome. Crossover probability is 1 and mutation probability is set to 1/186. The GA is terminated if there is no improvement in ten generations. A set of experiments showed that (100+10000) selection produces satisfactory results. The GA converged on an AMD Athlon™ XP 2000 machine within five minutes (within 40 generations).

Figure 11 illustrates the result of the automated clustering by the GA. There are five clusters (two in dark border, one in light border, one in dashed border, and one in shadows) and a bus. Two observations can be made as follows:

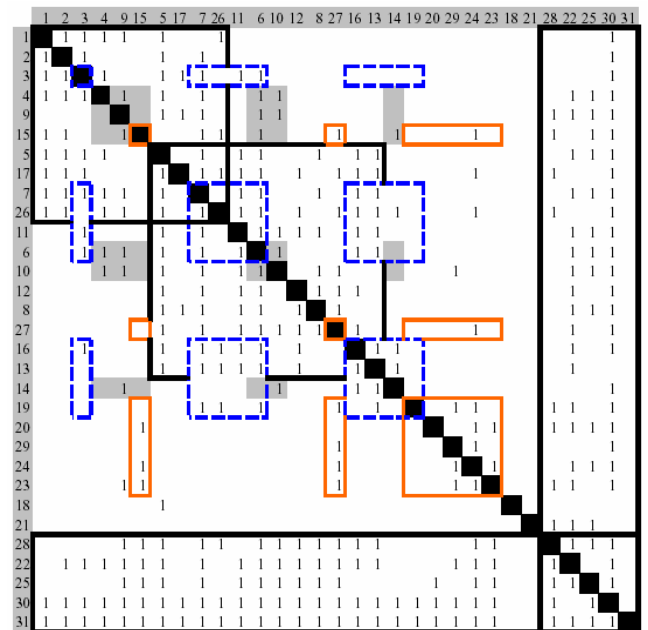


Figure 11: The clustering arrangement by the GA of the DSM of 10MWe gas turbine.

1. Manual clustering ignores three dimensional structures because DSMs are two dimensional representations. That

is, when humans are inspecting a DSM, they are really looking at a 2-dimensional projection of the real object (which may be 3-dimensional for complex products). On the contrary, the GA clustering is capable of finding three dimensional structures.

2. The GA clustering is more balanced in the two types of mismatches than the manual version. In the human version, some clusters are denser (e.g. Exhausts) while the others are sparser (e.g. Acoustic). In the GA version, clusters and the bus are roughly the same dense.

8.3 Discussion of Results

The result above can be interpreted by two points of view. If the MDL is a more appropriate criterion for the clustering problem, then the GA provides more rigorous solutions than humans. The description length of manual clustering in Figure 10 is 507.43 bits. It is superior compared to a random clustering which on average needs roughly 758 bits. However, the GA produced solutions which are on average roughly 406 bits. Figure 12 shows that the average solution quality given by the GA outperformed manual clustering after the 10th generations.

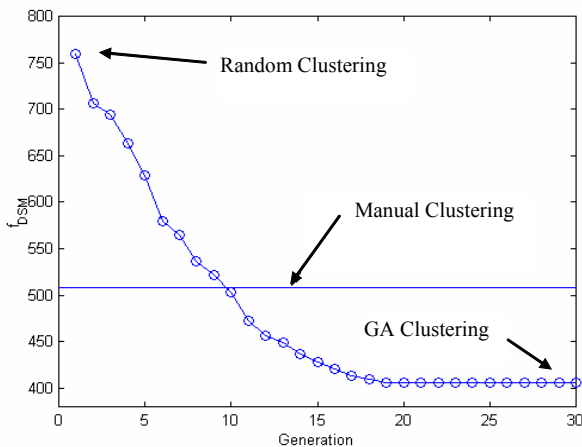


Figure 12: The description length of clustering given by human versus that given by the GA.

On the other hand, if human clustering is more appropriate due to this explanation in considering several subtle constraints which were not observed by the GA, then the problem is how to “tune” the MDL-metric to mimic human experts’ preference. As an initial attempt, we have done that by tuning α and β according to Figure 10 (as described in Section 8.2). However, there are still a number of possibilities for tuning the MDL-metric, such as tuning the weight of the model description. This is an area which requires further investigations.

9. CONCLUSIONS AND CONTINUING WORK

This paper started by reviewing the DSM literature related to clustering and product modularity. Then, we presented the MDL concept and used it as a metric for the clustering objective function. The MDL-based objective function was then applied to a GA to cluster DSMs. Several simplified illustrative examples showed that the GA is capable of solving DSM clustering problems with overlapping, a bus, and/or with a three dimensional structure. Finally, we applied the GA to a real-world problem—a 10MWe gas turbine. The result was compared with the manual clustering and to show the promise of automated clustering using GAs.

The DSM is a powerful tool for representing product architectures. This representation allows for the analysis and development of modular products by clustering the DSM. DSM clustering algorithms are scarce and inefficient especially when confronted with complex product architecture as in the gas turbine case study. The clustering algorithm presented in this paper was capable to identify complex clustering arrangements and overcome many of the difficulties in previous clustering tools. Using MDL as a principle to cluster DSMs is more systematic and seems promising. However, more research is still undergoing to refine our proposed clustering algorithm. These efforts include:

1. We would like to extend our work to non-zero-one domain, where the values of entries in the DSM represent the degree of dependency between two nodes.
2. The weights of the model description might be tuned in a similar manner. More experiments are needed to find the preference of human experts.

We believe that the algorithm presented in this paper serves as a strong basis for a series of rigorous mathematical DSM clustering algorithms. This should lead to improved products, processes, and organizational designs. Ironically, the MDL+DSM combination also lead us to investigate the dual problem of using DSM clustering to design more effective genetic algorithms [18].

REFERENCES

- [1] Ulrich, K., 1995, "The Role of Product Architecture in the Manufacturing Firm," *Research Policy*, 24, pp. 419-440.
- [2] Ulrich, K. and Eppinger, S., 2000, *Product Design and Development*, McGraw Hill, New York.
- [3] Baldwin, C. and Clark, K., 2000, *Design Rules: The Power of Modularity*, MIT Press, Cambridge.
- [4] McCord, K., Eppinger, S., 1993 "Managing the Integration Problem in Concurrent Engineering," Working Paper 3594, MIT Sloan School of Management, Cambridge, MA.
- [5] Pimpler, T., Eppinger, S.D., 1994, "Integration Analysis of Product Decompositions," Proceedings, ASME Design Theory and Methodology, DE-Vol. 68.

- [6] Fernandez, C., 1998, *Integration Analysis of Product Architecture to Support Effective Team Co-location*, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.
- [7] Thebeau, R., 2001, *Knowledge Management of System Interfaces and Interactions for Product Development Processes*, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.
- [8] Goldberg, D. E., 1989, *Genetic Algorithms in Search, optimization, and Machine Learning*, Addison-Wesley, New York, NY.
- [9] Rissinen, J., 1978, "Modeling by Shortest Data Description," *Automatica*, 14, pp. 465-471.
- [10] Lutz, R., 2002, "Recovering High-Level Structure of Software Systems Using a Minimum Description Length Principle," *Artificial Intelligence and Cognitive Science*, M. O'Neill, R.F.E. Sutcliffe, C. Ryan, M. Eaton, N.J.L. Griffith (Eds.): Proceedings of the 13th Irish International Conference, AICS 2002, Limerick, Ireland, Sept. 12, 02.
- [11] Alexander, C., 1964, *Notes on the Synthesis of Form*, Harvard Press, Boston, MA.
- [12] Hartigan, J., 1975, *Clustering Algorithms*, Wiley & Sons, New York, NY.
- [13] Sharman, D., Yassine, A., Carlile, P., 2002a, "Characterizing Modular Architectures," Proceedings, *ASME 14th International Conference*, DTM-34024, Montreal, Canada, Sept. 29 - Oct. 2, 2002.
- [14] Sharman, D., Yassine, A., Carlile, P., 2002b, "Architectural Optimisation Using real Options Theory and Dependency Structure Matrices," Proceedings, *ASME 28th Design Automation Conference*, DAC-34119, Montreal, Canada, Sept. 29 - Oct. 2, 2002.
- [15] Altus, S., Kroo, I., Gage, P., 1996, "A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems," *ASME Journal of Mechanical Design*, 118.
- [16] McCulley, C., and Bloebaum, C., 1996, "A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems," *Structural Optimization*, 12, pp. 186-201.
- [17] Sharman, D., 2002, *Valuing Architecture for Strategic Purposes*, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.
- [18] Yu, T.-L., Goldberg, D. E., Yassine, A., Chen, Y.-P., 2003, "A Genetic Algorithm Design Inspired by Organizational Theory: A Pilot Study of a Design Structure Matrix Driven Genetic Algorithm," IlliGAL Technical Reports, Publication No. 2003007.