

# DOUBLE DUTY: GENETIC ALGORITHMS FOR ORGANIZATIONAL DESIGN AND GENETIC ALGORITHMS INSPIRED BY ORGANIZATIONAL THEORY

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

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

## Abstract

Modularity is widely used in system analysis and design such as complex engineering products and their organization. Also, modularity is the key to solve optimization problems efficiently via problem decomposition. We first discover modularity in a system, and then leverage this knowledge to improve the performance of the system. In this chapter, we tackle both problems with the alliance of organizational theory and evolutionary computation. First, we cluster the dependency structure matrix (DSM) of a system using a simple genetic algorithm (GA) and an information theoretic based metric. Then we design a better GA through the decomposition of the optimization problem using the proposed DSM clustering method.

## 1 INTRODUCTION

Modularity is ubiquitous (Schilling, 2002). The concept can be found in and applied to many disciplines: biology (Hartwell *et al.*, 1999; Andrews, 1998), psychology (Fodor, 1996), social networks (Newman & Girvan, 2004; Guimerà *et al.*, 2003), engineering design (Ishii & Yang 2003; Fixson, 2003), engineering optimization (Altus *et al.*, 1996), and software development (Sullivan *et al.*, 2001), to name a few. Such modular structure in many of these natural and man-made complex systems is believed to improve the functionality, performance, and robustness of these systems (Alon, 2003; Baldwin & Clark, 2000). It allows increased product or organizational variety (Ulrich & Eppinger, 2000), increased rates of technological or social innovation (Baldwin & Clark, 2000); increased opportunities for market dominance through interface capture (Moore, 1999), and increased specialization at firm level which in turn may allow more flexible response to environmental change (Sanchez & Mahoney, 1996).

Therefore, revealing the modular structure of complex systems is crucial for developing performance enhancements techniques. In other words, we first have to discover modularity in a system, and then leverage this knowledge to improve the performance of the system. In this chapter, in particular, we first propose a genetic algorithm (GA) based method to discover the modular structure of a decomposed system (*e.g.* a product or organization), then we propose a new, improved genetic algorithm based on the modular arrangement of the system.

The proposed clustering method is developed based on a matrix representation of a graph (called the dependency structure matrix or DSM (Yassine & Braha, 2003)), the minimum description length (MDL) principle (Rissinen, 1978 and 1999; Barron *et al.*, 1998; Lutz, 2002), and a simple genetic algorithm (GA) (Goldberg, 1989). The method is capable of partitioning the product (or organizational) architecture into an *optimal* set of modules (or teams) and can be fine tuned to mimic clustering arrangements proposed by human experts. Our proposed clustering algorithm is an improvement over other existing algorithms for two reasons: (a) existing algorithms are manual, very dependent on human expertise, and consequently hard to automate or replicate (McCord & Eppinger, 1993; Pimmler & Eppinger, 1994; Stone *et al.*, 2000; Gonzalez-Zugasti *et al.*, 2000); (b) existing algorithms use simple mathematical constructs to discriminate between modules and consequently these algorithms collapse when confronted with complex product architectures (Fernandez, 1998; Thebeau, 2001; Whitfield *et al.*, 2002).

It is interesting that the clustering techniques can help us design better a GA, which decomposes the optimization problem by recognizing the modularity between decision variables. Holland (1975) has suggested that operators learning linkage information to recombine alleles might be necessary for GA success. Many such methods have now been developed to solve this including perturbation (Goldberg *et al.*, 1989; Goldberg *et al.*, 1993; Kargupta, 1996; Munetomo & Goldberg, 1999), linkage learning schemes (Harik & Goldberg, 1996; Smith, 2002), and model building schemes (Bosman & Theiren, 1999; Harik, 1999; Pelikan *et al.*, 1999). The genetic algorithm developed in this chapter adds to the literature of competent GAs with a method inspired by organizational theory. In particular, the proposed *dependency structure matrix genetic algorithm* (DSMGA) is able to identify BBs with the help of a DSM and accomplish BB-wise crossover. As the experimental results suggest, compared to a simple genetic algorithm, the DSMGA is able to maintain a reliable solution quality under tight, loose, and random linkage with the same amount of function evaluations.

The rest of the chapter proceeds as follows. The next section provides a quick overview of the DSM method. In Section 3, we propose a metric to evaluate different architectural arrangements based on the MDL principle. Section 4 describes the MDL-DSM clustering methods in detail and shows several illustrative examples. In Section 5, we demonstrate this new clustering method using a DSM for a 10 MWe gas turbine. In Section 6, we utilize this DSM clustering method to design a better genetic algorithm—DSMGA. Finally, Section 7 concludes the chapter.

## 2 THE DEPENDENCY STRUCTURE MATRIX (DSM) METHOD

A DSM is a matrix representation of a graph (Steward, 1981). The nodes of the graph (which represent components of a product or system) correspond to the column and row headings in the matrix (Eppinger *et al.*, 1994; Yassine & Braha, 2003). The arrows (which represent relationships between components) correspond to the “X” marks inside the matrix. 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 (Fernandez, 1998). As an example, consider the DSM in Figure 1. As can be seen in Figure 1(b), the original DSM was rearranged (by simply swapping the position of rows and columns) to contain most of the interactions within two separate blocks or modules: *AF* and *EDBCG*. However, three interactions are still outside any block. An alternative arrangement is suggested in Figure 1(c). This arrangement suggests the forming of two overlapping modules (*i.e.*, *AFE* and *EDBCG*).

The DSM representation of a system/product architecture have proved useful because of its visual appeal and simplicity and numerous researchers have used it to propose architectural improvements by simple manipulation of the order of rows and columns in the matrix (McCord & Eppinger, 1993; Pimmler &

Eppinger, 1994). In an attempt to automate this manual process of DSM inspection and manipulation, Fernandez (1998) used a DSM model with a simulated annealing search techniques in order to find “good” DSM clustering arrangements. In his approach, 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 and Yassine (2004) attempted using the clustering algorithm described in Fernandez (1998) on an industrial gas turbine. However, he showed that this algorithm is incapable of predicting 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 trapped in local optimal solutions. In a similar venue, Whitfield *et al.* (2002) used genetic algorithms to form product modules. Their algorithm is also built upon the same concepts introduced by Fernandez and as such suffers from similar problems.

The problem with applying automated clustering algorithms to 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 overlapping clusters, bus modules, and three-dimensional structures (Sharman & Yassine, 2004).

### **3 MINIMUM DESCRIPTION LENGTH (MDL) BASED METRIC**

The lack of an efficient clustering method particularly suited for analyzing product and organizational architectures motivated us to seek a better clustering metric based on information theoretic measures. Sharman and Yassine (2004) outlined the requirements necessary for the development of a new clustering metric and its corresponding 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 metric, as will be discussed in Section 4, the third requirement is directly related to the proposed search strategy and the encoding.

### 3.1 Model Description

Suppose that we have a model which describes a given product structure or a data set. Usually, the model does not completely describe the given data; otherwise, the model would become too complicated. Therefore, the description length needed to describe the whole 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 are 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 (Rissanen, 1978 and 1999; Barron *et al.*, 1998; Lutz 2002) satisfies our needs for dealing with the above tradeoff. 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.

### 3.2 Model Encoding

The way we encode the model is straightforward. 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 2 shows a DSM clustering arrangement and the corresponding model description. It is easily seen that the length of this model description is as follows:

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

where  $n_c$  is the number of clusters in the DSM,  $n_n$  is the number of nodes,  $cl_i$  is the number of nodes in the  $i^{\text{th}}$  cluster, and the logarithm base is 2. In the example of Figure 2,  $n_c = 2$  clusters,  $n_n = 8$  nodes,  $cl_1 = 3$ , and  $cl_2 = 4$ . 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.”

If  $n_n$  and  $n_c$  are known, it is not difficult to see that the above model description is uniquely decodable. When  $n_n$  is given, and assuming  $n_c \leq n_n$ , then  $\log n_n$  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.

### 3.3 Mismatched Data Description

Based on the model, we first construct another matrix (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. Define the following 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:

$$\sum_{(i,j) \in S_1} (\log n_n + \log n_n + 1) + \sum_{(i,j) \in S_2} (\log n_n + \log n_n + 1). \quad (2)$$

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.

### 3.4 MDL Clustering Metric

The MDL clustering metric is given by the weighted summation of the model description length given in Section 3.2 and the mismatched data description given in Section 3.3. With some arithmetic manipulations, the metric can be written as follows:

$$f_{DSM}(M) = (1 - \alpha - \beta) \cdot \left( n_c \log n_n + \log n_n \sum_{i=1}^{n_c} cl_i \right) + \alpha \cdot [ |S_1| (2 \log n_n + 1) ] + \beta \cdot [ |S_2| (2 \log n_n + 1) ], \quad (3)$$

where  $\alpha$  and  $\beta$  are weights between 0 and 1. Here we use weighting in order to match the preference of human experts. This becomes more evidential in the case study. The  $\alpha$  and  $\beta$  setting is not a simple task. A

naïve setting is  $\alpha = \beta = 1/3$ . In Section 4, we adjust  $\alpha$  and  $\beta$  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.

### 3.5 From Binary DSM to Weighted DSM

Most real-world DSMs are real valued or contain information of different levels of interaction strength. Therefore, it is necessary to extend our algorithm to be capable of clustering weighted DSMs. If we normalize a weighted DSM so that every entry in the DSM is between 0 and 1, the value of each entry can be considered as the probability of communication. This is consistent with binary DSMs. In a binary DSM,  $d_{ij} = 1$  can be thought as that node  $i$  communicates with node  $j$  with probability 1. The same interpretation is also valid for  $d_{ij} = 0$ . Based on the interpretation, we can modify the MDL scoring metric as follows. First, the entries  $d_{ij}$  in the DSM is normalized to  $p_{ij} = (d_{ij} - d_{\min}) / (d_{\max} - d_{\min})$ , where  $d_{\max} = \max_{i,j} d_{ij}$  and  $d_{\min} = \min_{i,j} d_{ij}$ . The formula of the description length of model complexity remains the same. The mismatch sets for type 1 and type 2 are modified as  $S_1 = \sum_{d_{ij}=1} (1 - p_{ij})$  and  $S_2 = \sum_{d_{ij}=0} p_{ij}$ , respectively. The modification is so because entry  $ij$  has a probability  $(1 - p_{ij})$  to be a type 1 mismatch if it is inside a cluster, and a probability  $p_{ij}$  to be a type 2 mismatch if it is outside clusters.

## 4 THE PROPOSED CLUSTERING ALGORITHM

To use a GA with the MDL clustering metric, an encoding method that encodes a clustering arrangement into a chromosome is needed. As indicated in Section 3, 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  $(n_c \cdot n_n)$  bits, where  $n_c$  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 2,  $n_n = 8$ , and given  $n_c$  is 3, then the model can be

described by the following chromosome shown in Figure 3. When manipulated, the chromosome is transformed into a binary string which is a concatenation of all rows.

With the above encoding scheme, now it is sufficient to apply the proposed MDL clustering metric to a GA to create a DSM clustering algorithm. At the beginning, a population of (encodings of) DSM clustering arrangements is randomly initialized. The MDL clustering metric is then applied to each DSM clustering arrangement, and the description length of each DSM clustering arrangement is obtained. With the evaluations, a GA with  $(\lambda+\mu)$  selection, uniform crossover, and simple mutation searches the DSM clustering arrangement with a minimal description length.

The parameters  $\alpha$  and  $\beta$  in Equation 3 can be tuned by the use of Widrow-Hoff iteration (or Delta rule) (Widrow & Hoff, 1960), so that the GA results would have similar ratios for the description length as human experts. For more details, refer to Yu *et al.* (2004).

## 5 CASE STUDIES

In this section, the proposed DSM clustering algorithm is tested on a real-world DSM for a 10 MWe gas turbine (Sharman & Yassine, 2004). The 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. Figure 4 shows an attempt at manually clustering the DSM (Sharman & Yassine, 2004). 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. Readers are referred to Yu, *et al.* (2004) for more real-world DSMs case studies.

### 5.1 Automated Clustering Using the Proposed MDL-GA

By inspecting several expert-clustered DSMs in Yu, *et al.* (2004), the average ratios of the description lengths of the model, type 1 mismatch, and type 2 mismatch is set as the average of 0.0784: 0.8116:0.1102. The maximal number of clusters is set to half of the number of nodes. Hence the length of the chromosomes for the DSM for the 10 MWe gas turbine is  $15 \times 31 = 465$ . Crossover probability is set to one, and mutation probability is set to one over the chromosome length. The GA is terminated if there is no improvement in fifty generations. A set of experiments showed that  $(4250+4250)$  selection produces satisfactory results. Since the number of Widrow-Hoff iterations is limited to 10, then after 10 iterations, the best run is chosen

according to the minimal sum of squared errors. The objective ratios and the experimental ratios obtained from the 10 Widrow-Hoff iterations are shown in Figure 5. Finally, applying the proposed MDL-GA clustering algorithm resulted in Figure 6.

## 5.2 Discussion of Results

Figures 7 and 8 show the clustering arrangements, mismatches, and description length performed by human experts manually and the proposed MDL-GA algorithm. According to Figure 8, the proposed MDL-GA clustered the gas turbine DSM with a less complex model and fewer mismatches, and the MDL-GA gave a shorter total description length. If the MDL-GA clusters the DSMs using the specific weights tuned for the individual case, then it will find a better arrangement (*i.e.*, less complex models and fewer mismatches), as depicted by the results of Figure 9. In this case, the MDL-GA gave shorter description lengths in all three categories: model, type 1 mismatch, and type 2 mismatch.

Furthermore, according to Figure 8 we can see that the magnitude of type 1 mismatch is always larger than type 2 mismatch. In other words, human experts tend to endure type 1 mismatch (*i.e.*, inside clusters) more than type 2 mismatch (*i.e.*, between clusters). This observation is intuitive if we consider that system engineers tend to pay more attention to minimizing interactions between clusters (*i.e.*, product modules or design teams) than interaction patterns inside a cluster. Therefore, an automated algorithm that would mimic human clustering would be biased in the same direction.

These results can be interpreted by two points of view. If the MDL is a more appropriate criterion for the clustering problem, then the GA provides better solutions than humans. On the other hand, if human clustering is more appropriate due to 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 accomplished that by tuning the weights ( $w_1$ ,  $w_2$ ,  $w_3$ ) according to the method described in Section 4. However, our point of view is that the MDL-GA need not provide better results than human expert clustering, but the ability to devise an automated algorithm capable of producing competitive clustering arrangements aligned with human expertise. The proposed method provides a consistent, systematic, and automatic way to cluster DSMs, and the clustering results can be either used directly, or used as an initial clustering arrangement for human experts to tune.

## 6 DEPENDENCY STRUCTURE MATRIX GENETIC ALGORITHM

The modularity among people in an organization suggests an optimal teaming, and hence minimizes the coordinate cost in the organization. The modularity among product components of a product line suggests an optimal parallelism, and hence maximizes the benefit for the product line. Similarly, the modularity among decision variables of an optimization problem suggests an optimal problem decomposition, and hence ensures the optimization problem can be solved *quickly, reliably, and accurately* (Goldberg, 2002).

In this section, we develop a technique for extracting building block (BB) information by the DSM clustering technique presented in previous sections, called a dependency structure matrix genetic algorithm (DSMGA) (Yu *et al.*, 2003). DSMGA is able to properly decompose the optimization problem with the help of DSM clustering, and then solves the problem by BB-wise crossover.

### 6.1 Descriptions of DSMGA

This section constructs DSMGA. The DSMGA consists of three main tasks: Constructing a DSM representing interactions between pairs of genes, clustering the DSM to obtain BB information, and perform BB-wise crossover by exploiting the BB information. This section describes the framework of the DSMGA, and how to accomplish the three tasks mentioned above.

#### 6.1.1 The Framework of the DSMGA

There are two levels of evolutionary algorithms in the DSMDGA: One is to solve the given problem, called the *primary GA*, and other one is to solve the building block (BB) identification problem, called the *auxiliary GA*. The basic idea of the DSMGA is to use the auxiliary GA to identify BBs, then the primary GA solves the optimization problem by utilizing the BB information.

Basically, the auxiliary GA receives a DSM which represents the dependencies between genes from the primary GA. After solving the DSM cluster problem, the auxiliary GA returns the BB information back to the primary GA, and the primary GA utilizes the BB information to accomplish BB-wise crossover. More details about the DSM construction and the auxiliary GA can be found in following subsections. Additionally, unlike a meta-GA (Mercer & Sampson, 1978), the auxiliary GA in the DSMGA does not consume function evaluations of the given problem. Instead, the auxiliary GA uses only the MDL clustering metric,  $f_{DSM}$  (Equation 3), as described in section 3.4, which is much less computationally expensive than many real-world objective functions.

### 6.1.2 DSM Construction

The way that the DSMGA detects the dependency of gene  $i$  and gene  $j$  is similar to LINC and LIMD (Munetomo & Goldberg, 1999). Define  $f(a_i = x, a_j = y)$  as the fitness value of the schema where the  $i$ -th gene is  $x$ , the  $j$ -th gene is  $y$ , and the rest are \* (wild card). For example, for  $i=2$  and  $j=5$  in a 5-bit problem,  $f(a_i = 0, a_j = 1) = f(*0**1)$ . If the  $i$ -th gene and the  $j$ -th gene are independent (linear),  $f(a_i = 0, a_j = 1) - f(a_i = 0, a_j = 0)$  and  $f(a_i = 1, a_j = 1) - f(a_i = 1, a_j = 0)$  should be the same. Therefore, the interaction (nonlinearity) between the  $i$ -th gene and the  $j$ -th gene is defined as following:

$$\Delta_{ij} = |f(a_i = 0, a_j = 1) - f(a_i = 0, a_j = 0) - f(a_i = 1, a_j = 1) + f(a_i = 1, a_j = 0)|. \quad (4)$$

However, the fitness value of schemata cannot be computed unless every possible combinations are visited. Now the task is to approximate  $\Delta_{ij}$  with  $s_{ij}$  which is computed based on the individuals seen so far. First, define the sampled fitness of a schema in the population of  $t$ -th generation as  $f(a_i = x, a_j = y)^t = \frac{1}{n_a} \sum_{a \in P_t, a_i=x, a_j=y} f(a)$ , where  $t$  is the generation,  $P_t$  is the population of the  $t$ -th generation,  $f$  is the fitness function,  $a$  is an individual where its  $i$ -th gene is  $x$  and  $j$ -th gene is  $y$ , and  $n_a$  is the number of such  $a$  in the population. We call that  $f(a_i = x, a_j = y)^t$  is *undefined* if  $n_a$  is zero. The information of interactions gathered from the population is  $s_{ij}^t = |f(a_i = 0, a_j = 1)^t - f(a_i = 0, a_j = 0)^t - f(a_i = 1, a_j = 1)^t + f(a_i = 1, a_j = 0)^t|$ .  $s_{ij}^t$  is *undefined* if any of the  $f(a_i, a_j)^t$  is *undefined*.

To utilize all individuals of all populations seen so far,  $s_{ij}^t$  is then averaged over generations. Defining a set  $D = \{s_{ij}^t \mid s_{ij}^t \text{ is defined.}\}$ , the average of  $s_{ij}^t$  is expressed as  $s_{ij}^t = \frac{1}{|D|} \sum_{t=1, s_{ij}^t \in D}^{t=T} s_{ij}^t$ , where  $T$  is the current generation. With a threshold  $\theta$ ,  $s_{ij}^t$  is then transferred into 0-1 domain, and a DSM is constructed:

$$d_{ij} = \begin{cases} 0, & \text{if } s_{ij} \leq \theta. \\ 1, & \text{if } s_{ij} > \theta. \end{cases} \quad (5)$$

The threshold is calculated by the  $k$ -mean algorithm (Hartigen & Wong, 1979) by setting  $k=2$ . The threshold can also be set according to some prior knowledge if available.

### 6.1.3 BB-wise Crossover

After the BB information is obtained from the auxiliary GA, BB-wise crossover can be achieved in the primary GA level. The BB-wise crossover is like an ordinary allele-wise crossover, but instead of mixing genes, the BB-wise crossover mixes BBs and will not disrupt BBs. The order of BBs is not significant because BBs have no (or little) interaction with each other by definition. For more details about BB-wise crossover, refer to Thieren and Goldberg (1994).

## 6.2 Empirical Results

The experiments were done by using a simple GA (SGA) as a test platform. The DSMGA tested here is a simple GA with the the DSM construction and an auxiliary GA with the MDL clustering metric. The test function is a 30-bit MaxTrap problem composed of 10, 3-bit trap functions. The 3-bit trap is given by

$$f_{trap^3} = \begin{cases} 0.9, & \text{if } u = 0 \\ 0.45, & \text{if } u = 1 \\ 0, & \text{if } u = 2 \\ 1.0, & \text{if } u = 3 \end{cases}, \quad (6)$$

where  $u$  is the number of 1's among the 3 bits.

Three linkage cases were tested: tight linkage, loose linkage, and random linkage. Define  $U(x)$  as a counting function which counts the number of 1's in  $x$ . In the tight linkage test, genes are arranged as  $fitness = f_{trap^3}(U(x_1, x_2, x_3)) + f_{trap^3}(U(x_4, x_5, x_6)) + f_{trap^3}(U(x_7, x_8, x_9)) + \dots$ . On the other hand, in the loose linkage test case, genes are arranged as  $fitness = f_{trap^3}(U(x_1, x_{11}, x_{21})) + f_{trap^3}(U(x_2, x_{12}, x_{22})) + f_{trap^3}(U(x_3, x_{13}, x_{23})) + \dots$ . In the random linkage test case, genes are simple arranged randomly. For more details about this test scheme, see Goldberg *et al.* (1989).

Given the failure rate to be 1/10, the population size is set as 182 by the gambler's ruin model (Harik *et al.*, 1997; Miller, 1997). In the primary GA, binary tournament selection was adopted, and no mutation was used. In the auxiliary GA, the maximal number of cluster  $n_c$  is 10 (equal to  $m$ ), and the mutation probability  $p_m$  was set to be 1/30. A  $(\lambda+\mu)$  selection was adopted, where  $\lambda=5$  and  $\mu=500$ .

Figure 10(a) shows the performance of the SGA using two-point crossover. The SGA worked only for the tight linkage case. For loose and random linkage cases, SGA did not work because of BB disruption. Correspondingly, Figure 10(b) illustrates the performance of the DSMGA using BB-wise two-point crossover. The DSMGA converged for all three tests. Even in the tight linkage test, the DSMGA (converged at the 22th generation) outperformed the SGA (converged at the 40th generation) because DSMGA disrupted fewer BBs. This argument can be verified by Figure 11, which shows the DSM created by the DSMGA for the tight linkage case. The perfect result is 10, 3-bit clusters located on the diagonal. As the figure indicates, at the fifth generation, the DSMGA were able to identify eight BBs; the DSMGA has successfully identified all BBs at the tenth generation.

## 7 SUMMARY AND CONCLUDING REMARKS

The chapter started with a delicious circle. We sought to discover modularity with the aid of a genetic algorithm, and at the same time we sought to improve genetic algorithms through the discovery of modularity. Although circular reasoning sometimes leads to logical inconsistency, here a stepwise process led to success on both counts. In this chapter, we first presented the concept of DSMs. Then we introduced the MDL concept and used it as a metric for the proposed clustering objective function. The MDL-based metric was then used with a simple GA to cluster weighted graphs or their corresponding DSMs. We applied the MDL-GA to a real-world problem—a DSM of a 10 MWe gas turbine. The results were compared to manual clustering to show the promise of automated clustering using GAs, and the parameters can be tuned to mimic expert clustering arrangements. Finally, the DSM clustering technique was utilized to develop a competent GA—DSMGA. A series of experiments showed that DSMGA are capable of properly decomposing the optimization problem and is much more powerful than a simple GA.

The DSM is a powerful tool for representing and visualizing product architectures. This representation allows for the analysis and development of modular products by clustering the DSM. Clustering algorithms are scarce and inefficient especially when confronted with complex product architectures as in the reported case studies. The MDL-GA clustering algorithm presented in this chapter was capable of identifying complex clustering arrangements and overcoming many of the difficulties observed in previous clustering tools. The proposed DSM clustering method has the following unique features: (1) it accounts for bus modules, (2) it allows overlapping modules, (3) it is specifically designed to overcome DSM manual/human clustering

problems, (4) it has a unique information theoretic clustering measure, (5) it has the tuning capability to mimic human experts' clustering, (6) it allows tuning of GA parameters for specific types of products by human experts (*i.e.*, different parameters/weights for different products), then the tuned algorithm (*i.e.*, particular weights) can be used repeatedly by others (*e.g.*, non-experts) for similar products, or future generations of the same product. As future work, several possible steps can be taken to further refine the proposed method. These efforts include an extension to multi-objective clustering, where the values of entries in the DSM represent different types of dependencies between two nodes (Schloegel *et al.*, 1999). Along similar lines, a more explicit representation of domain specific expert knowledge may allow for better tuning of the weights of the model description, and more experiments (*i.e.*, case studies) might be needed to find the real preference of human experts (Nascimento & Eades, 2001). Furthermore, the proposed method is capable of identifying buses and overlapped clusters, but other predominant architectural features may also need to be identified and incorporated into the MDL clustering metric. One such example is the concept of a mini-bus or a floating bus discussed in Sharman and Yassine (2004).

Interestingly, the MDL-DSM combination also led us to investigate the dual problem of using DSM clustering to design a more effective genetic algorithm—DSMGA (Yu *et al.*, 2003). DSMGA utilizes the DSM clustering technique to identify BBs. An MDL clustering metric used by the auxiliary GA was constructed for the DSM clustering problem. Empirical results have shown that using the BB information obtained by the DSM clustering technique can help the convergence of GAs on tight, loose, and random linkage problems. Just as the DSM clustering results can be reused in organization, the BB information obtained from DSM clustering can also be used in similar optimization problems. In other words, DSMGA creates a specific crossover operator for the given problem. For more information of such reusability, refer to Yu and Goldberg (2004). Furthermore, DSMGA can be readily extended to solve hierarchical problems (Pelikan & Goldberg, 2001), and also combined with principled efficiency-enhancement techniques such as fitness inheritance (Sastry *et al.*, 2004) and time continuation (Lima *et al.*, 2005).

There are many examples in this volume of nature inspired computing coming to the aid of management and commerce, and this chapter is certainly to be counted among their number. The use of a genetic algorithm and an MDL metric to improve our understanding of organizational and product design is no mean feat; however, this chapter took the additional step of using ideas from genetic algorithms and organizational theory to improve the toolkit of nature inspired computation itself.

We believe that this two-way street can be profitably exploited more often than it currently is because natural computation and organizations both are so intricately tied to successful adaptation and innovation. We invite other researchers to join this quest for both metaphorical and mechanical inspiration going forward.

## REFERENCES

- Andrews, J. (1998). Bacteria as Modular Organisms. *Annual Review of Microbiology*, 52, 105-126.
- Alon, U. (2003). Biological Networks: The Tinkerer as Engineer. *Science*, 301, 1866-1867.
- Altus, S., Kroo, I., and Gage, P. (1996). A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems. *Transactions of the ASME*, 118, 486-489.
- Baldwin, C., & Clark, K. (2000). *Design Rules: The Power of Modularity*, MIT Press, Cambridge.
- Barron, A., Rissenen, J., & Yu, B. (1998). The MDL Principle in Coding and Modeling, *IEEE Transactions Information Theory*, 44, 2743-2760.
- Bosman, P., & Thierens, D. (1999). Linkage Information Processing in Distribution Estimation Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, 1, 60-67.
- Eppinger, S. D., Whitney, D. E., Smith, R., & Gebala, D. (1994). A Model-based Method for Organizing Tasks in Product Development. *Research in Engineering Design*, 6(1), 1-13.
- Fernandez, C. (1998). *Integration Analysis of Product Architecture to Support Effective Team Co-location*, Master Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Fixson, S. (2003). *The Multiple Faces of Modularity—A Literature Analysis of a Product Concept for assembled Hardware Products*. Technical Report 03-05, Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI.
- Fodor, J. (1996). *The Modularity of Mind*. MIT Press, Cambridge, MA.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, NY.
- Goldberg, D.E., Korb, B., & Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 3, 493-530.
- Goldberg, D.E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56-64.
- Goldberg, D.E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

- Gonzalez-Zugasti, J., Otto, K. & Baker, J. (2000). A Method for Architecting product Platforms. *Research in Engineering Design*, 12, 61-72.
- Guimerà, R., Danon, L., Diaz-Guilera, A., Giral, F., & Arenas, A. (2003). Self-similar Community Structure in a network of Human Interactions. *Physical Review E*, 68, 065103(R).
- Harik, G., & Goldberg, D.E. (1996). Learning linkage. *Foundations of Genetic Algorithms 4*, 247-262.
- Harik, G., Cantú-Paz, E., Goldberg, D.E., & Miller, B.L. (1997). The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of populations. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, 7-12.
- Harik, G. (1999). Linkage Learning via Probabilistic Modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Urbana, IL.
- Hartigen, J.A., & Wong, M.A. (1979). A k-means clustering algorithm. *Applied Statistics* 28, 10–108.
- Hartwell, L., Hopfield, J., Leibler, S., & Murray, A. (1999). From Molecular to Modular Cell Biology. *Nature* 402, C47-C52.
- Holland, J.H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.
- Ishii, K., & Yang, T. (2003). Modularity: International Industry Benchmarking and Research Roadmap. *Proceedings of ASME Design Engineering Technical Conference 2003, DETC2003/DFM-48132*.
- Kargupta, H. (1996). The performance of the gene expression messy genetic algorithm on real test functions. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, 631-636.
- Lima, C. F., Sastry, K., Goldberg, D. E., & Lobo, F. G. (2005). Combining Competent Crossover and Mutation Operators: A Probabilistic Model Building Approach. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*. 735-742.
- Lutz, R. (2002). Recovering High-Level Structure of Software Systems Using a Minimum Description Length Principle. *Proceedings AICS-02, Lecture Notes in Artificial Intelligence*, 2464, 61-69, Springer-Verlag.
- McCord, K., & Eppinger, S. (1993). Managing the Integration Problem in Concurrent Engineering. Working Paper 3594, MIT Sloan School of Management, Cambridge, MA.
- Mercer, R.E., & Sampson, J.R. (1978). Adaptive Search Using a Reproductive Meta-plan. *Kybernetes* 7, 215-228.

- Miller, B.L. (1997). Noise, Sampling, and Efficient Genetic Algorithms. Doctoral Dissertation, University of Illinois at Urbana-Champaign, Urbana.
- Moore, G. (1999). Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers, New York, HarperBusiness.
- Munetomo, M., & Goldberg, D.E. (1999). Identifying Linkage Groups by Nonlinearity/Non-monotonicity Detection. Proceedings of the Genetic and Evolutionary Computation Conference 1999, 1, 433-440.
- Nascimento, H.A.D., & Eades, P. (2001). Interactive Graph Clustering Based Upon User Hints. Paper Presented at the Proceedings of the Second International Workshop on Soft Computing Applied to Software Engineering, Enschede, The Netherlands.
- Newman, M., Girvan, M. (2004). Finding and Evaluating Community Structure in Networks, Physical review E, 69.
- Pelikan, M., & Goldberg, D.E. (2001). Escaping Hierarchical Traps with Competent Genetic Algorithms. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), 511-518. Also IlliGAL Report No. 2000020.
- Pelikan, M., Goldberg, D.E., & Cantú-Paz, E. (1999). BOA: The Bayesian Optimization Algorithm. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99 I, 525-532.
- Pimpler, T., Eppinger, S.D. (1994). Integration Analysis of Product Decompositions. Proceedings of ASME Design Theory and Methodology, DTM '94, 343-351.
- Rissanen, J. (1978). Modeling by Shortest Data Description. Automatica, 14, 465-471.
- Rissanen, J. (1999). Hypothesis Selection and Testing by the MDL Principle. Computer Journal, 42, 260-269.
- Sanchez, R., & Mahoney, J. (1996). Modularity, Flexibility, and Knowledge Management in Product and Organization Design. Strategic Management Journal, 17, 63-76.
- Sastry, K., Pelikan, M., & Goldberg, D. E. (2004). Efficiency Enhancement of Genetic Algorithms via Building-Block-wise Fitness Estimation. Proceedings of the IEEE Conference on Evolutionary Computation, 1, 19-23.
- Schilling, M.A. (2002). Modularity in Multiple Disciplines. In Garud, R., Langlois, R., & Kumaraswamy, A. (eds) Managing in the Modular Age: Architectures, Networks and Organizations. Oxford, England: Blackwell Publishers, 203-214.

- Schloegel, K., Karypis, G., & Kumar, V. (1999). A New Algorithm for Multi-objective Graph Partitioning. Proceedings of the European Conference on Parallel Processing, 322-331, Toulouse, France, August/September 1999.
- Sharman, D., & Yassine, A. (2004). Characterizing Complex Product Architectures. Systems Engineering Journal, 7(1), 35-60.
- Smith, J. (2002). On Appropriate Adaptation Levels for the Learning of Gene Linkage. Journal of Genetic Programming and Evolvable Machines, 3, 129-155.
- Steward, D.V. (1981). The Design Structure System: A Method for Managing the Design of Complex Systems. IEEE Transactions on Engineering Management, 28, 77-74.
- Stone, R., Wood, K. & Crawford, R. (2000). A Heuristic Method for Identifying Modules for Product Architectures. Design Studies, 21(1), 5-31.
- Sullivan, K., Griswold, W., Cai, Y., and Hallen, B. (2001). The Structure and Value of Modularity in Software Design. ACM SIGSOFT Software Engineering Notes, 26 (5), 99-108.
- Thierens, D., & Goldberg, D. E. (1994). Convergence Models of Genetic Algorithm Selection Schemes. In Parallel Problem Solving from Nature, PPSN III, 119-129.
- Thebeau, R. (2001). Knowledge Management of System Interfaces and Interactions for Product Development Processes, Master Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Ulrich, K. & Eppinger, S. (2000). Product Design and Development, McGraw Hill, New York.
- Whitfield, R., Smith, J., & Duffy, A. (2002). Identifying Component Modules. Seventh International Conference on Artificial Intelligence in Design AID'02, Cambridge, UK, July 2002.
- Widrow, B., & Hoff, M. E., (1960). Adaptive Switching Circuits. 1960 IRE WESCON Convention Record, 4, 96-104. New York: IRE. Reprinted in Anderson and Rosenfeld, 1988.
- Yassine, A., & Braha, D. (2003). Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method. Concurrent Engineering Research & Applications, 11(3).
- Yu, T.-L., Yassine, A., & Goldberg, D.E. (2004). An Information Theoretic Method for Developing Modular Architectures using Genetic Algorithms. Research in Engineering Design (submitted).
- Yu, T.-L., Goldberg, D.E., Yassine, A., & Chen, Y.-P., (2003). Genetic Algorithm Design Inspired by Organizational Theory: Pilot Study of a Design Structure Matrix Driven Genetic Algorithm. Artificial Neural Networks in Engineering 2003 (ANNIE 2003), 327-332.

Yu, T.-L., & Goldberg, D.E. (2004). Dependency Structure Matrix Analysis: Off-line Utility of the Dependency Structure Matrix Genetic Algorithm. Proceedings of the Genetic and Evolutionary Computation Conference 2004 (GECCO 2004), 355-366.

## List of Figures

Figure 1: DSM clustering examples.

(a) Original DSM. (b) Clustered DSM. (c) Alternative Clustering.

Figure 2: Above is a clustering arrangement of a DSM. Below is the associated model description.

Figure 3: A chromosome that represents the model shown in the lower part of Figure 2. The binary string manipulated is 010100101010100100000000.

Figure 4: Manual clustering of the gas turbine with named clusters (The numbers (1, 2, and 3) in the figure represent varying dependency strengths).

Figure 5: Results of the Widrow-Hoff iterations. The first row is the objective ratios of description lengths.

The second row shows the resulted ratios and the weights that produce such ratios.

Figure 6: Clustering arrangement by the proposed GA for the gas turbine DSM

Figure 7: Comparisons of the clustering arrangements given by human experts and the proposed GA using the averaged weights.

Figure 8: Description length and mismatches of the DSM clustering arrangement done by human experts versus by GAs.

Figure 9: Comparisons of the clustering arrangements given by human experts and the proposed GA using the weights according to the respective human clustering arrangement. It is worth noting that the results given by the GA dominate in all three categories.

Figure 10: (a) The performance of the SGA with two-point crossover. (b) The performance of the DSMDGA with BB-wise two-point crossover.

Figure 11: The DSMs created by the DSMDGA in the tight linkage test. From the left to the right, the DSMs are created at generation 0, 5, and 10, respectively. The perfect result should be 10, 3-bit clusters on the diagonal.

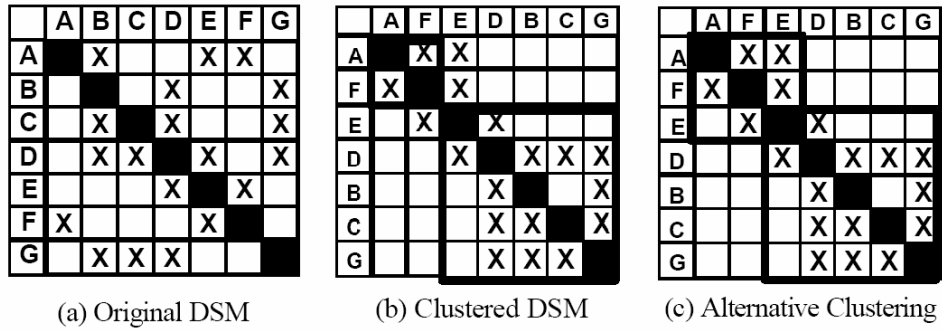


Figure 1: DSM clustering examples.

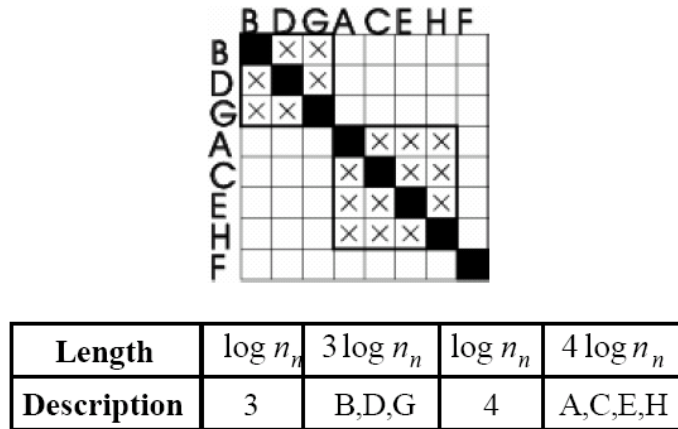


Figure 2: Above is a clustering arrangement of a DSM. Below is the associated model description.

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

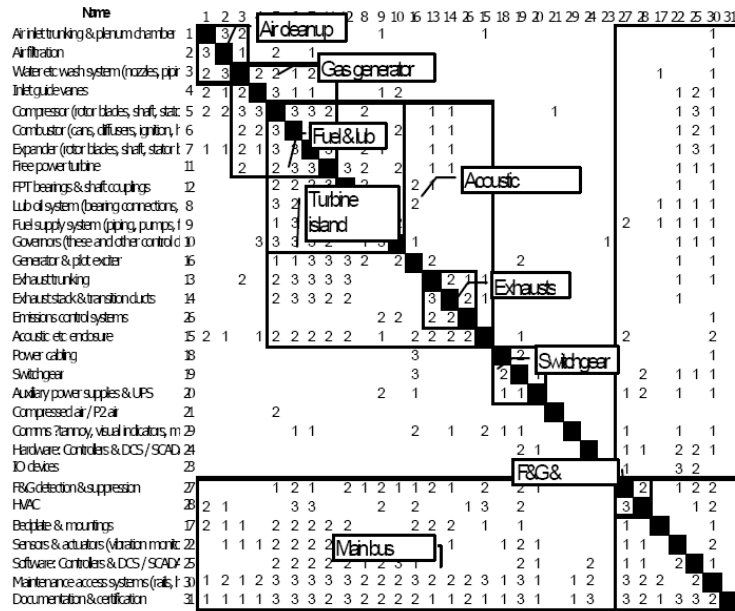


Figure 4: Manual clustering of the gas turbine with named clusters (The numbers (1, 2, and 3) in the figure represent varying dependency strengths)

	$w_1$	$w_2$	$w_3$	Description length ratios		
<b>Objective</b>				0.0784	0.8116	0.1102
<b>Results</b>	0.4533	0.1228	0.4239	0.0729	0.8154	0.1117

Figure 5: Results of the Widrow-Hoff iterations. The first row is the objective ratios of description lengths. The second row shows the resulted ratios and the weights that produce such ratios.

	1	2	3	4	10	6	11	13	14	26	15	17	12	27	6'	11'	8	9	10'	16	22	25	31	19	20	24	27'	28	18	21	29	23	5	7	30							
Air inlet trunking & plenum chamber	1	1	3	2							1								1																	1						
Air filtration	2	3	2	1																																2	1	1				
Water etc wash system	3	2	3	3	2		1	3				1																									2	2	1			
Inlet guide vanes	4	2	1	2	4	2	1												1		1	2														3	1	1				
Govermas	10				3	1	3	2																										1		3	3	1				
Combustor	6		2	2	2	6	3	1	1																												3	3	1			
Free power turbine	11		2	2	2	3	11	1	1			3																									2	3	1			
Exhaust trunking	13		2			3	3	13	2	1	1	3										1																2	3	1		
Exhaust stack & transition ducts	14					3	2	3	14	2	1	2										1																2	3			
Emissions control systems	26				2		2	2	26										2																				1			
Acoustic etc enclosure	15	2	1	1		2	2	2	2	2	15	2	2						1	2			1														2	2	2			
Bedplate & mountings	17	2	1	1		2	2	2	2	1	17	2	1							2																		2	2	1		
FTP bearing & shaft couplings	12					2	3	1		3	3		12		2	3	2			2	1																	2	2	1		
F&G detection & suppression	27					2	2	2	1	2		2	27		2		1	2	1	1	1	2																	1	1	2	
Combustor	6'														6'	3			2	1	2																	3	3	1		
Free power turbine	11'													3	3	11'	2		2	1	1																	2	3	1		
Lub oil system	8									1	3	2	3	8	3			2	1	1																			3	3	1	
Fuel supply system	9									1	3	2	3	3	3	9	2			1	1																		1	3	1	
Govermas	10'														3	2	1	3	10'	1	1	1																	3	3	1	
Generator & pilot exiter	16					2							3	1	3	2		2	16	1			2															1	3	1		
Sensors & Actuators	22	1	1	1			1						3	1	2	2	3	3	3	22			2	1	1	1	1											2	2	2		
Software: controllers & DCS/SCAD	25												2	1	2	2	1	2	3	1	2	25			2	1	2	1	1										2	2	1	
Documentation & certification	31	1	1	1	1		1	1	2	1	1		3	3	3	2	2	2	2	3	3	31		3	1	3	3	2	1	1								3	3	2		
Switch gear	19																						3	1	1		19	1											2	2		
Auxiliary power supplies & UPS	20																		2		1	1		1	20		1	2	1											1		
Hardware: controllers & DCS/SCAD	24																					2	2	2	1	24	1	1												1		
F&G detection & suppression	27																					1	1	2	2	1	27	2											1	1	2	
HVAC	28	2	1		3			1	3						2					2		2	1	2		3	28												3	2		
Power cabling	18																					3																	18		1	
Compressed air/P2 air	21																																						21		2	
Comms Tannoys, visual indicator	29					1		1	2			1							2	1			1																29		1	1
IO devices	23												1										3	2																23		
Compressor	5	2	2	3	3		3	2	1	1					3	2	2					1	3																5	3	1	
Expander	7	1	1	2	1		3	3	1	1					3	3	2	1				1	3																3	7	1	
Maintainness access systems	30	1	2	1	2	2	3	3	2	2	2	3	2	3	3	3	2	2	2	3		2			3	1	2	3	2	1	1							3	3	30		

Figure 6: Clustering arrangement by the proposed GA for the gas turbine DSM

	$n_c$	$cl_i$	$ S_1 $	$ S_2 $
<b>Manual</b>	8	2, 3, 3, 3, 6, 7, 8, 13	342.33	17.67
<b>GA</b>	6	3, 3, 5, 9, 9, 11	233.67	32.00

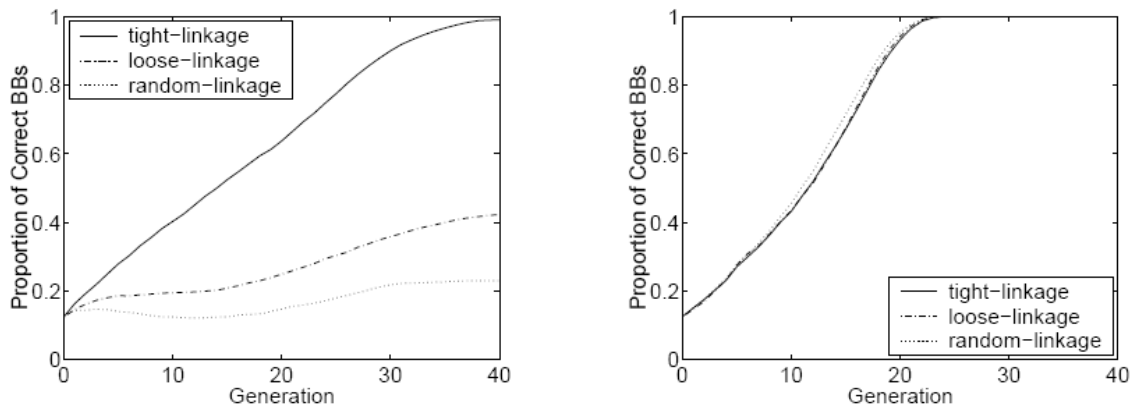
Figure 7: Comparisons of the clustering arrangements given by human experts and the proposed GA.

Description length	Model	Type 1 mismatch	Type 2 mismatch	Total Weighted Description Length
<b>Manual</b>	262.57	3897.93	192.71	3205.38
<b>GA</b>	227.89	2548.93	349.07	2125.05

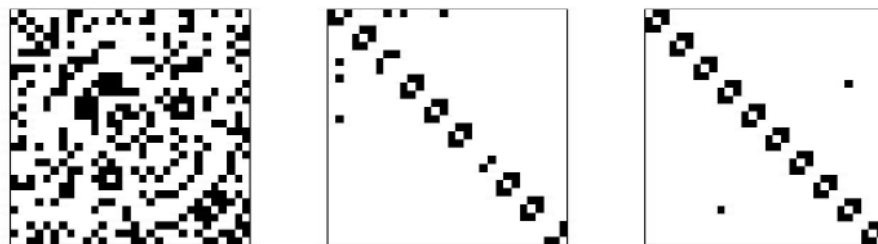
Figure 8: Description length and mismatches of the DSM clustering arrangement done by human experts versus by GAs.

Description length	Model	Type 1 mismatch	Type 2 mismatch
Manual	262.57	3897.93	192.71
GA	208.72	3421.60	174.53

**Figure 9: Comparisons of the clustering arrangements given by human experts and the proposed GA using the weights according to the respective human clustering arrangement. It is worth noting that the results given by the GA dominate in all three categories.**



**Figure 10: (a) The performance of the SGA with two-point crossover. (b) The performance of the DSMDGA with BB-wise two-point crossover.**



**Figure 11: The DSMs created by the DSMDGA in the tight linkage test. From the left to the right, the DSMs are created at generation 0, 5, and 10, respectively. The perfect result should be 10, 3-bit clusters on the diagonal.**