Proceedings of DETC'05
2005 ASME Design Engineering Technical Conferences
Long Beach, California, USA, September 24-28, 2005

DETC2005/DTM-85173

HIERARCHICAL MODULARITY: DECOMPOSITION OF FUNCTION STRUCTURES WITH THE MINIMAL DESCRIPTION LENGTH PRINCIPLE

Bingwen Wang and Erik K. Antonsson, Ph.D., P.E.*
Engineering Design Research Laboratory
California Institute of Technology
1200 E. California Blvd.
Pasadena, CA 91125-4400
bingwen@caltech.edu, erik@design.caltech.edu

ABSTRACT

In engineering design and analysis, complex systems often need to be decomposed into a hierarchical combination of different simple subsystems. It's necessary to provide formal, computable methods to hierarchically decompose complex structures. Since graph structures are commonly used as modeling methods in engineering practice, this paper presents a method to hierarchically decompose graph structures. The Minimal Description Length (MDL) principle is introduced as a measure to compare different decompositions. The best hierarchical decomposition is searched by evolutionary computation methods with newly defined crossover and mutation operators of tree structures. The results on abstract graph without attributes and a real function structure show that the technique is promising.

1 INTRODUCTION

It is common in engineering practice to hierarchically decompose large complex systems into small subsystems. Modular structures are often used to make the development of large systems more manageable and understandable. In engineering design synthesis, modular structures have many of advantages, including reducing cost [18], increasing flexibility [1, 18, 23], boosting the rate of innovation [1]. A particularly challenging situation is in redesign processes, where a configuration already exists and it is required to redesign the architecture of products to make them more efficient and more modular.

Due to complexity, many systems are modeled with abstract representations, including bondgraphs [7] in the analysis of dynamical systems, and function structures [2, 7, 8] in engineering design. The function representations [12] are extensively used in engineering design. In these graph representation, every node models a basic function unit, and links represent the interactions between different units including control signals, data information, material and power flow, etc. An example of this type graph representation is shown in Figure 1. In engineering modeling, it is common to associate the links of graphs with attributes representing the strength and types(domains) of the interactions. For example, the interactions could be energy, signal, and/or material.

A previous paper [27] by the authors introduced a method for identifying and measuring modularity of graph structures based on Shannon's measure of information complexity [19]. The method is based on the Minimal Description Length (MDL) principle building on observations of the strong (inverse) relation between complexity and modularity. In this paper, the method is extended and verified to decompose complex systems into *hierarchical* modular configurations. This paper will discuss how an abstract graph can be decomposed into a hierarchically modular structure and apply the method to function structures.

Prior related work relating to modularity in design includes [3, 4, 12, 20–22, 28]. Pahl, *et al.*, develops the function

* Address all correspondence to this author.

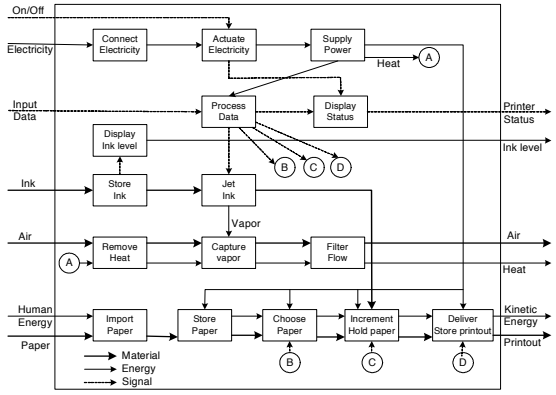

Figure 1. GRAPH REPRESENTATION OF A FUNCTION STRUCTURE. It's a function structure of an HP 1200C desktop inkjet printer. This figure is from K. Otto and K. Wood's Product Design [11, page 372].

structure representation and also discuss the modularity on function structures. Stone, *et al*., [22] develops a set of three heuristic methods for identifying modules from function structures derived based on a function basis and time ordered function chains. Yassine, *et al*., [20, 28], Sosale [21] and Gershenson [3, 4] propose different measures of modularity on design structure matrices.

2 MINIMAL DESCRIPTION LENGTH PRINCIPLE

The primary motivation for relying on MDL is the inductive inference of a general hypothesis for given data [14–17, 24–26]. In the practice of science and engineering, it is common to develop inferences in two steps:

1. Propose possible hypotheses based on the observed phenomena and data.
2. Evaluate the hypotheses and select one according to an objective measure of how well it models the observed data.

A common approach to selecting a hypothesis or theory is Occam's Razor principle, which is both intuitively appealing and informally applied throughout the sciences. This principle indicates that in general one should pursue the simplest hypothesis which gives a good prediction. The MDL principle is a form of Occam's Razor. The basic idea behind MDL is that any regularity in the data can be used to compress the data, since it takes shorter message to describe the regularity than the length of the message needed to describe the data literally. Based on this approach, MDL must represent the regularities in the data and that information which cannot be represented by the regularities. That is, MDL represents data as a string S consisting of two parts. The first part S_H encodes a hypothesis, and the second part S_D encodes the data based on the hypothesis with an efficient coding method, i.e., $S = S_H : S_D$. The principle states that the best hypothesis is the one giving the shortest description of the data based on the hypothesis. MDL chooses a hypothesis that trades-off between how well it fits the observed data, and the complexity of the hypothesis.

To formalize the MDL principle, it is necessary to provide a description, that is, a formal language, to express regularities and properties of the data. The MDL principle depends on the particular language or representation used. However, it has been shown that for any two general languages, the description lengths differ by no more than a constant c, which is the so-called invariance theorem. That is, as long as the sequence is long enough, it is not critical which general language or representation is chosen. The idealized MDL is general and powerful from a theoretic viewpoint, but is not computable in its general form .

In practice, MDL employs more restrictive languages, not general ones. However, the restrictive language used must be able to describe most regularities, although it could miss some. Another difficulty met in the application of MDL is to encode data in an efficient way. Fortunately, it is not necessary to construct an encoding, instead it is only necessary to calculate the length of the encoding for any hypothesis with sufficient accuracy, and then to find a hypothesis which minimizes the approximate length. The practical applications of MDL use such approximations [25]. The general approach is to choose an encoding scheme which provides reasonably compact encodings of the data, and yet is not too complicated to decode. This gives a good approximation to the true complexity of the objects being analyzed.

3 MDL, DECOMPOSITION and MODULARITY

In order to hierarchically decompose an abstract graph, it is necessary to provide a measure for comparing different decompositions. Intuitively, a modular system requires that components in a module are highly integrated and interactions across modules are minimized. Mapped to abstract graph models, hierarchical decomposition minimizes the connection between modules and makes the nodes inside modules connected as strongly as possible. A module in a decomposition can be thought of as a cluster whose elements are strongly connected among themselves and relatively weakly connected to elements in other units. A measure for different hierarchical decompositions should be able to to quantify coupling and integration. The MDL principle has been previously proposed by the authors as one such measure [27].

Complex systems, which usually have low regularity, are likely to have modular (decomposable) structures, and on the other hand modular structures can simplify the complex systems and make them easier to understand. Modularity can be viewed as a kind of regularity inside the complex systems. With the MDL principle, different hierarchical decompositions can be viewed as different *hypotheses*, and the abstract graph as *data* waiting to be described by the decompositions. A measure of

decomposition can be considered to be the minimal description length of the messages describing a system under a decomposition. The MDL principle doesn't consider coupling and integration separately but views them together. Weak coupling between modules means that most of the function units inside modules should not interact with other units outside the module. That is, only a small number of components inside each module should be known to other modules, and most of the contents of modules are invisible to other modules and only visible to other units in the same module. Therefore, a small number of descriptive elements (an alphabet) is needed to describe (or encode) a modular system. Where a high level of integration exists, units in the same module have much shared common information compared to units in different modules. By putting those units into a module, naming the module and encoding shared common information in the module, some of the information in the unit can be obtained by reference to the module name, and therefore the encoding of the message is shortened. Compared to the information (message) of a non-modular structure, less information (a shorter message) is required to describe a modular structure.

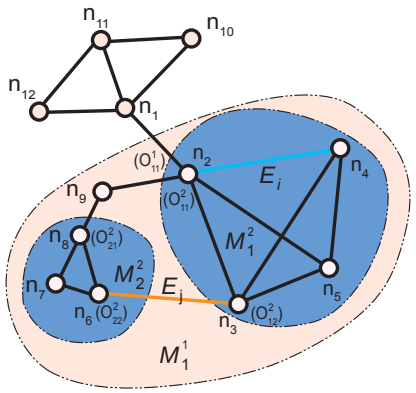

Figure 2. ABSTRACT GRAPH REPRESENTATION OF AN EXAMPLE STRUCTURE. Nodes in the structure (corresponding to individual functions in a function structure) are labelled n_1 through n_{12}. Link E_i represents an interaction between nodes n_2 and n_4. The boundaries of the modules are shown with the double-dot-dashed lines. The larger region labelled M_1^1 is a module, and the smaller blue regions labelled M_1^2 and M_2^2 are submodules. Interface O_{11}^1 is the connection between submodule M_1^2 and node n_1. Link E_j represents an interaction between modules M_1^2 and M_2^2.

4 ENCODING GRAPH STRUCTURES

What is the information in a graph? It includes nodes, links and attributed associated to links, and could also include interfaces. In modular structures, interfaces separate modules from other parts of the system, and hide most of the information inside modules. Interfaces should be a part of modules, and should be considered while describing modular structures. In dealing with abstract graphs, the interfaces are interpreted as the nodes of modules which have links to nodes outside the modules.

In messages describing graph structures, the following information should be recorded:

1. Units inside modules, including nodes and submodules.
2. Links which connect different units including nodes and submodules.
3. Interfaces through which units inside modules interact with other units outside modules.

The message format used to encode graph structures is as follows:

1. The whole graph is a unit
2. Unit = Name tables + list of Links
3. Name tables = Names of units and interfaces which are visible in the level.
4. Link = Name of two vertex units + attributes.
5. Name of vertex units could be ⟨Node⟩ or ⟨Submodule⟩ ⟨Interface in Submodule⟩.

For instance, link E_i in Figure 2 can be represented as "$n_2 n_4$⟨attribute values⟩". The nodes may be those inside submodules, *e.g.*, link E_j in Figure 2 can be represented as "$M_2^2\ O_{22}^2\ M_1^2\ O_{12}^2$ ⟨attribute values⟩".

An example message for the decomposition of the graph structure shown in Figure 2 is presented below:

M_1^1 (Module name) O_{11}^1 (List of Interfaces) {
  M_1^2(Module name) $O_{11}^2 O_{12}^2$(List of Interfaces) {
    $n_2\ n_3$ (links)
    $n_2\ n_4$
    $n_2\ n_5$
    $n_3\ n_4$
    $n_3\ n_5$
    $n_4\ n_5$}
  $M_2^2\ O_{21}^2\ O_{22}^2$ {
    $n_6\ n_7$
    $n_6\ n_8$
    $n_7\ n_8$}
  $n_9\ M_1^2\ O_{11}^2$
  $n_9\ M_2^2\ O_{21}^2$
  $M_2^2\ O_{22}^2\ M_1^2\ O_{12}^2$}
$n_1\ n_{10}$
$n_1\ n_{12}$
$n_{12}\ n_{11}$
$n_{11}\ n_{10}$
$n_1\ M_1^1\ O_{11}^1$

4.1 Names and Links

The names of units are only used to distinguish those units from each other; they have no physical meaning. Names could be represented by any code which has a unique name for each of the different units. The message of a module includes the information of the interfaces and the information inside the module, but not the the information of submodules. That is, the submodules are treated as a whole and the information inside them is hidden from the module while calculating the information of the module.

Let:

$L_j^{(n)}$ be the name length of node j.
$L_j^{(m)}$ be the name length of submodule j.
$L_{jk}^{(o)}$ be the name length of interface k of submodule j.

Then the message length for names of links (name tables) in the module is:

$$\begin{aligned} I^{(nk)} &= \text{Sum of the lengths of all links in the module} \\ &= \sum_{j=1}^{N^{(n)}} L_j^{(n)} \times N_j^{(n)} + \sum_{j=1}^{N^{(m)}} L_j^{(m)} \times N_j^{(m)} \\ &\quad + \sum_{j=1}^{N^{(m)}} \left(\sum_{k=1}^{N_j^{(o)}} L_{jk}^{(o)} \times N_{jk}^{(o)} \right) \end{aligned} \tag{1}$$

where:

$N^{(n)}$: number of single nodes
$N_j^{(n)}$: number of links connected to node j
$N^{(m)}$: number of submodules
$N_j^{(m)}$: number of links connected submodule j
$N_j^{(o)}$: number of interfaces of submodule j
$N_{jk}^{(o)}$: number of links connected to interface k of submodule j

With this information, it is possible to calculate the encoding length for nodes, submodules and interfaces in submodules. Within a module, nodes and submodules are at the same level, so it is applicable to encode them with the same name table. If one unit j has N_j connections to other units inside the same module, then the encoding for this unit will be used N_j times in the total coding used to describe links inside the module. From an information theoretic view, the message length could be minimized if the more frequently occurring names are assigned to the shorter codes. In accordance with Shannon coding [19], the optimal length of the label, which has frequency of p_j, should be $-\log(p_j)$. In our problem, p_j can be estimated as $N_j / \sum_{j=1}^{N_i^{(u)}} N_j$.

Then the total encoding length related to unit labels is:

$$-\sum_{j=1}^{N_i^{(u)}} N_j \log(p_j) = -\sum_{j=1}^{N_i^{(u)}} N_j \log\left(N_j / \sum_{j=1}^{N_i^{(u)}} N_j \right) \tag{2}$$

The message length at a specific level is the summation of message lengths of modules at this level. Then, it's easy to get the total length of messages needed to encode names in links at level n as:

$$\begin{aligned} I_n^{(n)} &= \sum_{\text{all modules at level } n} I^{(nk)} \\ &= -\sum_{\text{all}} \sum_{j=1}^{N_i^{(u)}} N_j \log\left(N_j / \sum_{j=1}^{N_i^{(u)}} N_j \right) - \\ &\quad \sum_{\text{all}} \sum_{j=1}^{N^{(m)}} \left(\sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)} \log\left(N_{jk}^{(o)} / \sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)} \right) \right) \end{aligned} \tag{3}$$

4.2 Attributes

According to whether there exists order between different values of attributes, attributes can be separated into two categories: unordered attributes and ordered attributes. Generally unordered attributes are multi-state and used to represent class index, such as domain attribute in mechanical system. Ordered attributes are usually used to represent physical quantities, and can be discrete (such as power level in digital system) and continuous (such as power level in analogy system). Here unordered attributes and discrete ordered attributes can be encoded by the same strategy and ordered attributes by another different strategy. In the following both unordered attributes and discrete ordered attributes are called multi-state attributes.

4.2.1 Multi-state Attributes In this case, the techniques used for encoding names can be applied. Let t_j be a value for T_j, and N_{t_j} be the number of links having value t_j of multi-state attribute T_j. The relative frequency of occurrence of state t_j of attribute T_j in the module will be estimated by:

$$p_{t_j} = \frac{N_{t_j}}{\sum_{t_j} N_{t_j}} \tag{4}$$

Suppose there are N_T multi-state attributes in a module, then the message length used to encode multi-state attributes in the

module is:

$$I^{(T_i)} = -\sum_{j=1}^{N_T}\sum_{t_j} N_{t_j}\log(p_{t_j}) \tag{5}$$

The total message length used to encode multi-state attributes at level n is:

$$I_n^{(mA)} = -\sum_{\text{all modules at level } n}\sum_{j=1}^{N_T}\sum_{t_j} N_{t_j}\log(P_{t_j}) \tag{6}$$

4.2.2 Continuous Ordered Attributes Unlike multi-state attributes, usually there is precision associated with continuous ordered attributes since in engineering there is generally some tolerance accompanying measurements. Let an ordered attribute be A_i and corresponding precision be δ_i. In order to encode the information of those attributes, the attributes are first discretized by $A_i \leftarrow \lfloor A_i/\delta_i + 0.5 \rfloor$, then the message length for integer A_i could be $L_{A_i} = \log(A_i) + 2\log(\log(A_i))$ [10]. The total message length used to encode those ordered attributes at level n is:

$$I_n^{oA} = \sum_{\text{all attributes at level } n} L_{A_i} \tag{7}$$

4.3 Total Message Length

Since decompositions are hierarchical, modular structures at different levels have different effects on the whole structure. The overall message length I can be aggregated from the lengths of messages at different levels (represented as I_n), which are defined as follows: level n is set to 0 for the top level and the lower the module in the hierarchy, the larger n. If a linear aggregation is used, the total message length I is:

$$I = \sum_n C(n)\left(I_n^{(n)} + I_n^{mA} + I_n^{oA}\right) \tag{8}$$

5 COMPUTATION

Given a graph structure, the process of partitioning the graph into different hierarchical clusters involves searches in a large space with a complicated landscape. Garey [13] has shown that a k-way graph partitioning problem, which splits a weighted undirected graph into k clusters, is NP-complete. One suitable way to search such a difficult space is a genetic algorithm, whose search procedure is based on the mechanism of natural selection. Genetic algorithms were introduced in [6], and were subsequently developed by Goldberg *et. al.*, [5].

5.1 Algorithm Setup

In order to use genetic algorithms, the following elements must be provided:

1. *Encoding schemes*. Since GA's work with a coding of the parameter set, not on parameters themselves, an encoding scheme is required. The scheme used in this paper is discussed in Section 5.2.
2. *Population initialization*. A common way is to randomly and uniformly sample the search space, which is used in the following experiments.
3. *Genetic operators*. The population is evolved with mutation and crossover operators. The operators are discussed further in Section 5.3.
4. *Evaluation Function*. A measure is used to tell how fit individuals are as solutions to the problem. Since here the best modular structures of abstract graphs are being sought, an information measure of modular structures is used as an indicator of fitness. The fitness of genome x is defined to be:

$$\text{The Fitness } f(x) = \frac{N}{I(x)} \tag{9}$$

where $I(x)$ is the information measure of the modular structure encoded as genome x, and N is the number of the nodes in the graph structure.
5. *Selection Mechanisms*. In genetic algorithms, selection occurs in two ways. One is how parents are selected to reproduce offspring, and the other is how individuals are selected from this generation and their offspring to form the next generation. There are many different selection mechanisms such as rank-based, roulette wheel selection, tournament selection, and deterministic selection [5]. Here roulette wheel selection is used to select parents, and deterministic selection for generational selection. One important parameter related to the second kind of selection is the generation gap: the percentage of new individuals in the new generation. For example, the generation gap of a simple genetic algorithm is 1.00, and steady state genetic algorithms have a low generation gap. In steady state genetic algorithms, only a fraction of weakest individuals will be replaced by offspring. In this paper, steady state genetic algorithms are used and 1% of the individuals were replaced in every generation.

5.2 Encoding Scheme

As shown above, partitioning is used to cluster the nodes in the graph. The encoding only records which nodes belong to each cluster. It is unnecessary to encode links. An obvious way to represent the hierarchical relation between the different nodes is using tree structures, in which every graph node appears once on the leaves of the tree representation and every leave represents

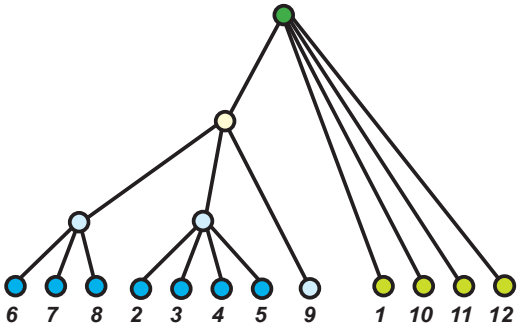

Figure 3. TREE REPRESENTATION OF HIERARCHICALLY MODULAR STRUCTURES

one graph node. For example, the modular structure shown in Figure 2 could be represented as shown in Figure 3.

5.3 Genetic Operators

Using general genetic operators, there is no guarantee that legal genome structures will be produced. Two approaches can be used to deal with this problem. One is to insert a penalty in the fitness function for illegal structures, and the other is to design new operators which only produce legal genomes. The later method is used here. Operators for tree structures have been developed previously [9]. However, those methods can't guarantee that the resulting tree structures are legal in the sense that every graph node appears once on the leaves of the tree representation and every leaf of the tree represents one graph node. It is necessary to design new genetic operators that guarantee that legal tree structures are produced.

5.3.1 Crossover The crossover operator includes five steps.

1. Randomly select two parents (T_1, T_2) by roulette wheel method. Two parents are shown in Figure 4(a).
2. Uniformly randomly select two hidden nodes from the two parent trees respectively, label the two subtrees under the selected nodes as DT_1, DT_2 and the two leftover subtrees as LT_1, LT_2, and make copies (CT_1, CT_2) of DT_1, DT_2, as shown in Figure 4(a).
3. Keep subtrees LT_1, LT_2 unchanged, delete common leaves of CT_1 and CT_2 from T_1 and T_2, and delete the uncommon leaves from CT_1 and CT_2, as shown in Figure 4(b).
4. Randomly select a hidden node from (T_1 - LT_1) and add subtree CT_2. Similarly for T_2 and CT_1, as shown in Figure 4(c).
5. Clearance. The hidden nodes that have no child nodes will be removed. Those hidden nodes having only one child node will be removed and its child node will be attached to its parent, as shown in Figure 4(d).

5.3.2 Mutation There are four different mutation operators: swapping two leaves, swapping two subtrees, merging leaves, and splitting a large subtree.

1. *Swapping two leaves:* First, randomly select two leaves from different clusters and then exchange them, as shown in Figure 5(a).
2. *Swapping two subtrees:* First randomly select two subtrees, neither of which is a subtree of the other, and then exchange them, as shown in Figure 5(b).
3. *Merging leaves:* Randomly select a subtree, and merge the leaves according to the following cases:

   CASE 1: If the subtree has few single nodes (say 1 or 2), those single nodes will be merged into a randomly selected subtree under the subtree, as shown in Figure 5(c).
   CASE 2: If the subtree has many single nodes, all those nodes will be merged into a cluster under the subtree, as shown in Figure 5(d).
   CASE 3: If one submodule in the selected subtree has very few single nodes, the submodule will be merged into another submodule with the least leaves among the remaining submodules, as shown in Figure 5(e).

4. *Splitting a large subtree:* If a tree has many subtrees, it will be split into two subtrees, as shown in Figure 5(f)

6 Applications

In application, it is usually necessary to map a graph structure into an abstract graph which can be accessed by computers. For example, the function structure shown in Figure 1 can be presented as the abstract graph in Figure 9(a). With an abstract graph, the method hierarchically decomposes the nodes into a modular structure, performs a pre-measure process, and then calculates the message length. The decomposition step and the message length calculation have been discussed above. The pre-measure step is strongly related to the interfaces, which play an important role in modular structures. If one module M has many of the same kind of interactions with other modules, it is possible to create a single output port on module M and establish connections so that other modules interact with M through the output port. As shown in Figure 6, module M_2 has four type t interactions with module M_1, so a new output port P is built in module M_2. Components A, B, C, D do not directly interact with module M_1, but do so through output port P. It is also noted that in Figure 6 module M_2 has two type s interactions with M_1. Should this also be a candidate for the creation of an additional output port? It is necessary to establish rules to tell whether it is beneficial to create an output port. One way to do this is to compare the description lengths of the structures with or without the output port. In general, the description lengths of all structures

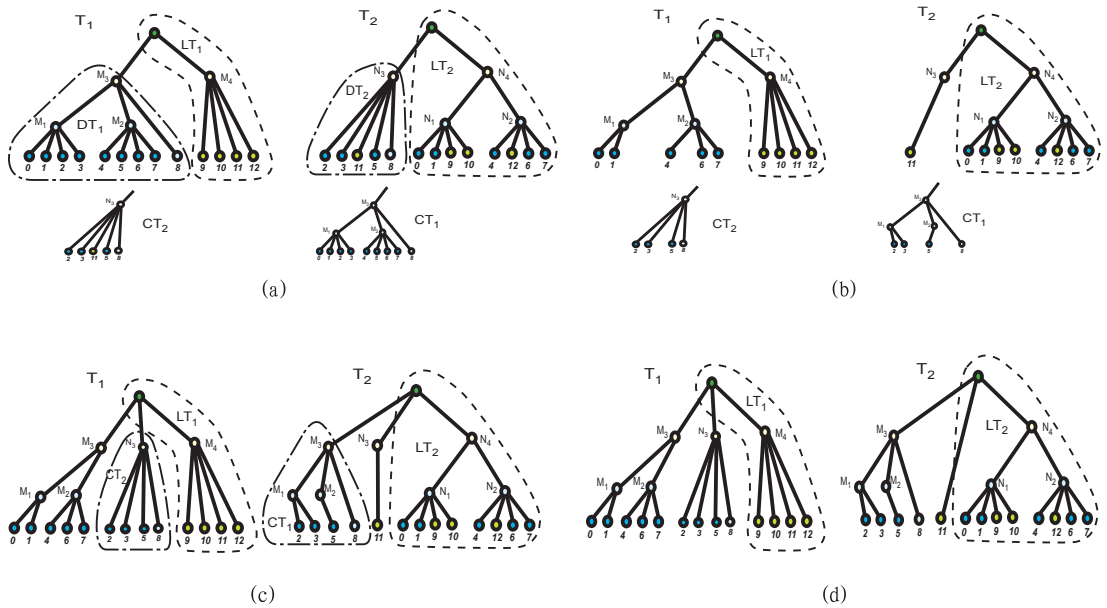

Figure 4. CROSSOVER OPERATOR: (a) T_1 and T_2 are two genome structures of initial parents, DT_1, DT_2 are randomly selected subtrees, and CT_1, CT_2 are copies of DT_1, DT_2; (b)Deleting repeated leaves; (c)Attaching the leftover CT_1 and CT_2 to T_1 and T_2 respectively; (d) Clearance: deleting N_2 node which has one leaf.

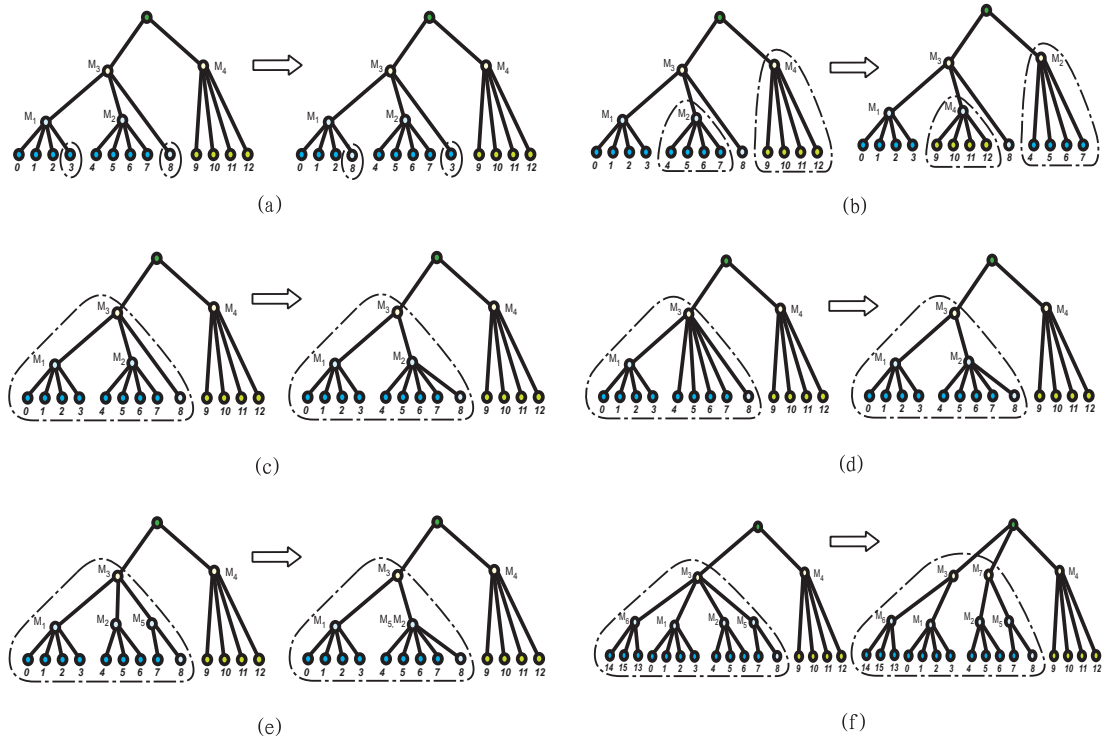

Figure 5. MUTATION OPERATORS: (a)Swapping tow leaves; (b)Swapping two subtrees; (c)Merging single nodes into other subtrees; (d)Merging single nodes as a subtree; (e)Merging two subtrees; (f) Splitting a subtree.

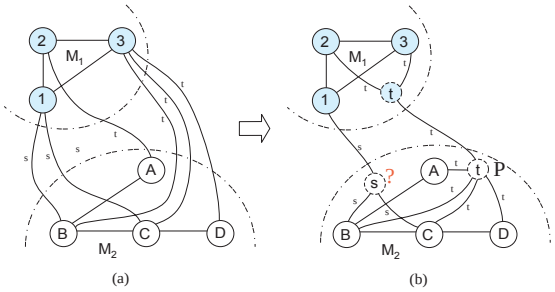

Figure 6. PREMEASURE: (a) The original graph; (b) One configuration after one premeasure process.

with different output port configurations should be calculated. The structure with the shortest length will be the best configuration of the output ports. However, in engineering practice, for some kinds of interactions, especially those related to geometry, it is not feasible to apply this technique. An example might be an output shaft from a power transmission whose purpose is to convey shaft power to multiple locations. In those cases, the premeasure step will not be applied.

In the following, the decomposition method is applied to an abstract graph without attributes, and a function structure.

6.1 An Abstract Graph Without Attributes

The raw graph with no identified modularity is shown in Figure 7(a). In this genetic computation, the population size is set to 150, the crossover rate to 0.9, the mutation rate to 0.3, and the aggregation function $C(n)$ is set to be $C(n) = 0.5 * (n_{\max} - n) + 1$. The aggregation function is used to balance the significance of the modularity at different levels. If the aggregation emphasizes the lower levels too much, the measure will favor the structures with fewer modules at lower levels. In this problem there is no preference for modularity at different levels, so $C(n)$ is set to be a constant C. The early computation shows the method gives the same decompositions while C varies from 0 to 1.0. That is, the decomposition results are not sensitive to the aggregation function. Further study on the effects of different aggregations is still needed.

Figure 7(b) shows a modular structure found by the algorithm. The convergence of the computation is shown in Figure 8. It should be noted that due to the stochastic nature of the algorithm, convergence to a modular decomposition is not always achieved.

6.2 Function Structures

The function structure of HP1200C printers is shown in Figure 1 [11]. The abstract graph representation is shown in Figure 9(a). In the genetic computation, the population size is set to 100, the crossover rate to 0.9, the mutation rate to 0.3, and the aggregation function $C(n)$ is set to be $C(n) = 0.5 * (n_{\max} - n) + 1$.

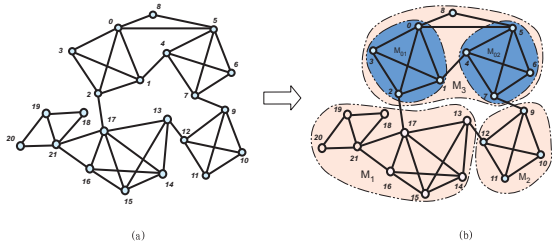

Figure 7. ABSTRACT GRAPH DECOMPOSITION: (a) The original graph; (b) One decomposition found by the algorithm.

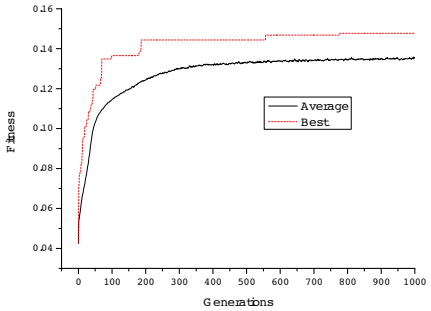

Figure 8. THE CONVERGENCE OF GA COMPUTATION FOR ABSTRACT GRAPH DECOMPOSITION.

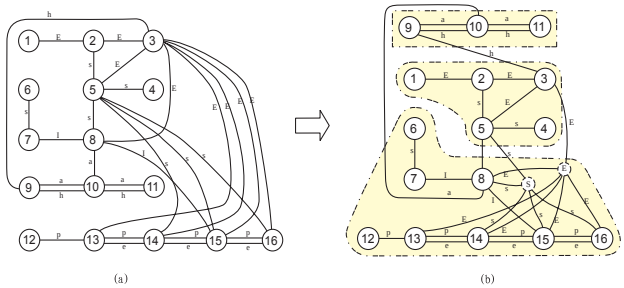

Figure 9. FUNCTION STRUCTURE DECOMPOSITION: (a) The abstract graph representation of the function structure of HP1200C. E: power; e: human energy; s: signal; p: paper; I: ink; h: heat; a: air. (b) One decomposition found by the technique.

Figure 7(b) shows a modular decomposition found by the algorithm. The convergence of computation is shown in Figure 10.

The result obtained by the technique in this paper, shown in Figure 9(b), is nearly identical to the modular decomposition [11] as shown in Figure 11. Components $6, 7, 8, 11, 12, 13, 14$ are put into one module in our result while they are separated into two modules $6, 7, 8$ and $11, 12, 13, 14$ in Figure 11. By inspection alone, it is difficult to tell which decomposition is better. However, in practice concerns not represented by the function structure, including geometry, may play an important role in ranking the relative benefits of several possible

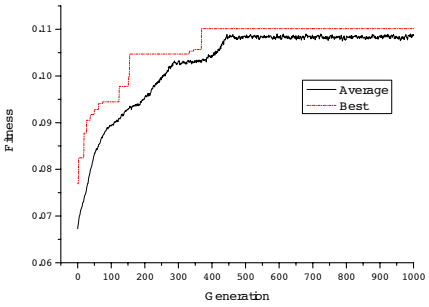

Figure 10. THE CONVERGENCE OF GA COMPUTATION FOR FUNCTION STRUCTURE DECOMPOSITION.

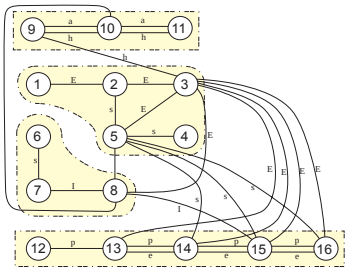

Figure 11. THE MODULAR DECOMPOSITION OF THE FUNCTION STRUCTURE FROM [11].

modular decompositions.

7 CONCLUSIONS AND FUTURE WORK

In this paper, an information based measure of modularity is used to hierarchically decompose abstract graphs representing engineering function structures. New mutation and crossover operators have been developed to stochastically search for hierarchical decompositions. These initial results show that the message description length principle (MDL) is an applicable measure for comparing hierarchical decompositions. The use of MDL in a genetic algorithm fitness function can produce modular decompositions of engineering function structures that compare favorably with those produced manually.

The work described above shows that the technique is promising, but still needs evaluation on more complex systems. Continuing work includes finding the optimal values for the various GA parameters, and to investigate the scalability and computational complexity of the technique of the computation over a range of graph size and complexity. Attributes of engineering designs beyond those represented by function structures will be incorporated into this approach in the future.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for valuable comments that improved the presentation of this paper.

REFERENCES

[1] BALDWIN, C. Y., AND CLARK, K. B. *Design Rules, Volume 1, The Power of Modularity*. MIT Press, Cambridge MA., 2000.

[2] FENVES, S., AND BRANIN, F. Network topological formulation of structural analysis. *ASCE, Journal of The Structural Division 89*, 4 (1963), 483–514.

[3] GERSHENSON, J., PRASAD, G., AND ZHANG, Y. Product modularity: Measures and methods. *Journal of Engineering Design 15*, 1 (2004), 33–51.

[4] GERSHENSON, J. K., PRASAD, G. J., AND ALLAMNENI, S. Modular product design: A life-cycle view. *Journal of Integrated Design and Process Science 3*, 4 (1999).

[5] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization,and Machine Learning*. Addison-Wesley, MA., 1989.

[6] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1975.

[7] KARNOPP, D., AND ROSENBERG, R. *System Dynamics: a Unified Approach*. John wiley & sons, 1975.

[8] KAVEH, A. Graphs and structures. *Computers and Structures 40* (1991), 893–901.

[9] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selections*. MIT Press, 1992.

[10] MING, L., AND VITANYI, P. M. B. *An Introduction to Kolmogorov Complexity and Its Application*. Springer-Verlag, New York, 1993.

[11] OTTO, K., AND WOOD, K. *Product Design*. Prentice Hall, 2001.

[12] PAHL, G., AND BEITZ, W. *Engineering Design: A Systematic Approach*. Springer, 1996.

[13] R.GAREY, M., AND JOHNSON, D. S. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co., 1979.

[14] RISSANEN, J. Modeling by shortest data description. *Automatica 14* (1978), 465–471.

[15] RISSANEN, J. A universal prior for integers and estimation by minimum description length. *Annals of Statistics 11* (1983), 416–431.

[16] RISSANEN, J. Stochastic complexity and modeling. *Annals of Statistics 14* (1986), 1080–1100.

[17] RISSANEN, J. Hypothesis selection and testing by the MDL principle. *Computer Journal 42* (1999), 260–269.

[18] SANCHEZ, R., AND MAHONEY, J. T. Modularity, flexibility, and knowledge management in product and organ-

isation design. *Strategic Management Journal 17 (Winter Special Issue)* (1996), 63–76.

[19] SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal 27* (July and October 1948), 379–423 and 623–656.

[20] SHARMAN, D. M., YASSINE, A. A., AND CARLILE, P. Architectural optimisation using real options theory and dependency structure matrices. In *Proceedings of the ASME DETC DAC 2002* (2002).

[21] SOSALE, S., HASHEMIAN, M., AND GU, P. An integrated modular design methodology for life-cycle engineering. *Annals of the CIRP 46* (January 1997).

[22] STONE, R., WOOD, K., AND CRAWFORD, R. A heuristic method to identify modules from a functional description of a product. In *Proceedings of DETC98* (Atlanta, GA, 1998).

[23] ULRICH, K., AND TUNG, K. Fundamentals of product modularity. In *ASME Issues in Design/Manufacture integration* (Atlanta, USA, 1991), pp. 73–80.

[24] WALLACE, C. S., AND BOULTON, D. M. An information theoretic measure for classification. *Computer Journal 11* (1968), 185–194.

[25] WALLACE, C. S., AND DOWE, D. L. Minimum message length and Kolmogorov complexity. *Computer Journal 42* (1999), 270–283.

[26] WALLACE, C. S., AND FREEMAN, P. R. Estimation and inference by compact coding (with discussion). *J.R. Statist Soc., B 49* (1987), 240–265.

[27] WANG, B., AND ANTONSSON, E. K. Information Measure for Modularity in Engineering Design. In *Proceedings of DETC'04: 2004 ASME Design Engineering Technical Conferences* (Salt Lake City, UT, Sept. 2004). Paper Number DTM-57515.

[28] YU, T.-L., YASSINE, A., AND GOLDBERG, D. A genetic algorithm for developing modular product architectures. In *Proceedings of the ASME DETC DTM 2003* (2003).