Proceedings of Proceedings of DETC'04
2004 ASME 2004 Design Engineering Technical Conferences
Salt Lake City, Utah, USA, September 28-October 2, 2004

DETC2004/DTM-57515

INFORMATION MEASURE FOR MODULARITY IN ENGINEERING DESIGN

Bingwen Wang
Engineering Design Research Laboratory
California Institute of Technology
1200 E. California Blvd.
Pasadena, CA 91125-4400
bingwen@caltech.edu

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

ABSTRACT

Modular structures are common in complex natural and artificial systems, and the terms "modular" or "modularity" are used throughout the engineering design literature. However, formal ways to measure or quantify modularity are still needed. This paper introduces an information-based approach to measure modularity, built on the relationship between complexity and modularity. In this information-based measure, a modular structure is encoded as a message describing information contained in the modular structure; the shorter the message, the higher the modularity of the structure. The information measure is dependent on the modeling and representation of the system. Following this basic idea, an approximate expression for the information measure of abstract graph structures is introduced. Since function structures in engineering design are typically represented as abstract graphs, this approach can be used to synthesize favorable modularity in parallel with the design of new systems. Using a genetic algorithm approach, with the reciprocal of the approximate measure as the fitness function, modular configurations are found in abstract graphs.

1 INTRODUCTION

Modularity is a concept that has proved useful in a large number of complex natural or artificial systems [1, 23, 25, 28], and modular structures have many advantages [1, 27], which include simplified assembly and disassembly, shortened development time and easier maintenance. Pahl and Beitz [16] pointed out that in mechanical system design processes, if modularity is identified and exploited in the initial conceptual or reverse engineering effort, development time and costs for product design will be significantly reduced. Due to the importance and utility of modular structures, this topic has recently received considerable interest in many different fields [1, 20, 28]. This interest has produced a multiplicity of definitions of modularity and some measures for modularity [6, 22, 32]. A metric to compare different clustering of design structure matrices (DSM) is provided in [22, 32]. This paper introduces a measure of modularity based on information theory, after a brief discussion of the relation between complexity and modularity. A specific information measure of modularity for graph structures is presented, followed by a verification using a genetic algorithm for a sample graph.

2 MODULARITY

2.1 What Is Modularity?

Modularity is an attribute of a system describing the degree of overall coupling between function units. This coupling may occur at multiple length-scales and dimensions. Modularity has following characteristics:

1. Hierarchy. A modular system can be separated into subsystems, and subsystems can also be separated into subsubsystems. Modularity of the whole system can be defined and measured, and it should also be possible isolate the subsystems and define and measure their modularity.
2. Globality. Modularity is a global characteristic of a system and is an integration of the modularity of its subsystems.

*Address all correspondence to this author.

3. Multi-Dimensionality. Modularity is related to many different aspects such as physical structure, logical structure, and temporal relationships.
4. Functionality. Modularity should be strongly associated with the functions of a system.
5. Granularity. When developing a modular system, it is important to consider the size of the modules. The greatest benefits of a highly modular systems are produced with a level of granularity that is neither overly fine, nor overly coarse.
6. Universality. A useful definition of modularity should work across a broad range of cases.

To measure modularity, the following questions need to be answered:

What is a formal way to quantify coupling?

What are the formal ways to measure the levels of the function units in function hierarchy and granularity of modules?

How can modularity of multiple subsystems in different dimensions be aggregated into an overall measure of modularity?

2.2 Complex vs. Modular

Modular structures are common in natural systems, and are highly associated to complex systems. As discussed by Simon in "The Architecture of Artificial" [23], complex systems from business organizations to biological systems exhibit a property of being "nearly decomposable", *i.e.*, modular structures. For instance, a human body is a complex and modular system consisting of many different morphological units, which are also complex and hierarchically modular.

Modularity is intrinsic and beneficial to complex systems [19], and may be a general principle for managing complexity. Modular architectures are used to design complex artificial systems, and to establish models to analyze complex natural and social systems. By breaking up a complex system into discrete modules or building a complex system using modules, it is possible to clarify complicated systemic interconnections and therefore render complexity manageable. In engineering, increasingly large teams are needed to design an artifact as it becomes more and more complex. It is necessary to divide the knowledge and effort needed in the design process into multiple smaller units, and to provide mechanisms to coordinate interactions between these modules.

Modular architectures make complex systems easier to understand and manage, and may play a critical role in the synthesis of complex systems [28].

2.3 Information-based Measure of Complexity

Modularity can make complexity easier to manage, so measures of complexity can provide some clues to measures of modularity. There are many methods to measure complexity depending on different views of complexity, such as Bennett's [3] logical depth, Gellmann's [8] total information, and information-based complexity [4, 14, 24, 26, 30]. Among them, an information-based method measure is well-known and commonly used. The study and applications of information-theoretic measure of complexity are based on work originated by Kolmogorov [14] and Solonmonoff [24]. Kolmogorov complexity $K_U(S)$ of a finite string is the length of the shortest program that computes on a universal Turing machine. Theoretically Kolmogorov complexity is perfect, but it is non-computable. It is necessary to find a good[1] approximation to $K_U(S)$ for applications. One good approximation is Minimum Description Length/Minimum Message Length (MDL/MML), which was first introduced by Wallace and Boulton [29] in clustering analysis, and refined by Wallace and Freeman [31], and Rissanen [17, 18]. MDL/MML is a variant of Occam's Razor principle, or the Principle of Parsimony: the simpler the explanation, the better; one should not make more assumptions than the minimum needed.

The primary motivation for MDL/MML is the inductive inference of a general hypothesis for given data. Like Kolmogorov complexity, it also 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 data based on the hypothesis specified in the first part with an efficient encoding method. *i.e.*, $S = S_H : S_D$. The principle states that a simple hypothesis gives a short length of the string encoded on the basis of the hypothesis. Similar to Occam's Razor principle, the MDL/MML method is to find a hypothesis which gives the shortest string length as the model or theory of data. Additional details on the MDL/MML method, can be found in [2, 10]. The most significant difficulty in applying this principle is to encode data in an efficient way. Fortunately, it is not actually necessary to construct an encoding, instead is it only necessary to calculate with sufficient accuracy the length of the encoding for any hypothesis, and then to find a hypothesis which minimizes the approximate length. The practical applications of MML use such approximations [30].

2.4 Information-based Measure of Modularity

The close relation between modularity and complexity suggests that an information measure be introduced into the measure of modularity. A modular structure of a complex system can be viewed as an encoded message, which describes information inside the modular structures. Information could include many aspects. The information used to describe two systems realizing different function sets are totally different. The more functions a system realizes, the more information will be needed to describe it. In engineering design, it is common to be interested in comparing the modularity of different realizations or architectures for

[1] "Good" means that the approximation should be computable and still describe complexity of a body of information as Kolmogorov complexity does.

the same design goals. Therefore it is reasonable to view modularity as a relative quantity, in the sense that the comparisons happen between available modular structures that provide similar functions.

As an analogy to MML method, those different architectures or designs for a specific task can be viewed as different "hypothesis", and the system is "data" waiting to be described under the design. A measure of modularity can be considered as minimal description/message length of messages describing a system under the modular structures. The shorter the code length of the message encoding a modular structure, the less information is inside the structure, which is necessary to understand and manage it, and the less complex and the more modular it is. Modularization can be regarded as a process to limit the information contained in the structures.

Now an information measure can be evaluated regarding its ability to address the three questions for quantifying modularity: quantifying coupling, granularity, and aggregation. The strength of coupling between two modules can be viewed as how much information flows between them, that is, how much the information contained within individual modules will be exposed to other modules. Weak coupling between modules means that most of the function units inside modules should not contact other units outside the module. That is, only a small amount of components, units or attributes 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. Introducing interfaces make hidden information feasible.

Compared to the information of a whole integrated structure, less information is required to describe modular structures. A smaller number of descriptive elements (an alphabet) is needed to describe (or encode) a modular system. A module focuses on one or a few essential functions, and units in the same module have more 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 shorten the encoding of the message. A larger module will need a longer message to encode the information inside the module.

Modular structures are hierarchical, in which the whole system is represented as the root or trunk, and modules or submodules are represented as nodes or leaves. Levels of modules or submodules can be defined as the depths of the corresponding nodes. The level of the root is assumed to be 1. From an engineering design point of view, while designing a modular system, the system designer should know the information flowing between different modules and the interfaces, and the module designer should know the information hidden inside the module and also the interfaces. Therefore, the overall information can be thought as the aggregation of the information at different levels. The information at a specific level is the summation (linear aggregation) of the information of every module at this level, and the information of a module includes the information of interfaces and information inside the module, but not the information of submodules, this 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. There are many different ways to aggregate information at different levels (represented as I_n) into the overall information. However, the modularity of subsystems at lower levels dominates those at higher levels. This is consistent with practical intuition, in that the information/decisions at the system level are more important than the information/decisions at module levels. This puts a special requirement on aggregation functions $f(I_1,\cdots,I_n)$:

$$\text{If } i > j, \text{ then } 0 < \frac{\partial f(I_1,\cdots,I_n)}{\partial I_i} \leq \frac{\partial f(I_1,\cdots,I_n)}{\partial I_j}$$

One special case is linear aggregation. A function $C(n)$ of level n can be used to aggregate information at different levels, then the total information I is:

$$I = \sum_n C(n) I_n \tag{1}$$

and $C(n)$ should be a decreasing function of n whether it is linear or nonlinear in n.

In design synthesis or analysis, the system is modelled in some framework, and formalized by some representation languages, as shown in Figure 1. Under this agreed framework, the required information can be defined.

To further explain information measure of modularity, a specific information measure for modularity of graph structures is implemented in next section. Abstract graph structures are commonly used in engineering as modeling methods [5, 12, 13], for example, a function structure in mechanical system [16] or software engineering can be abstracted as a graph $G(V,E)$, as shown in Figure 2. Every basic function unit can be modelled as a node, and interactions between those nodes as edges in graph $G(V,E)$.

3 INFORMATION MEASURE FOR GRAPH STRUCTURES

In messages describing graph structures, the following things 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. In graph representation, interfaces are interpreted as the nodes of modules, which have links to nodes outside the modules.

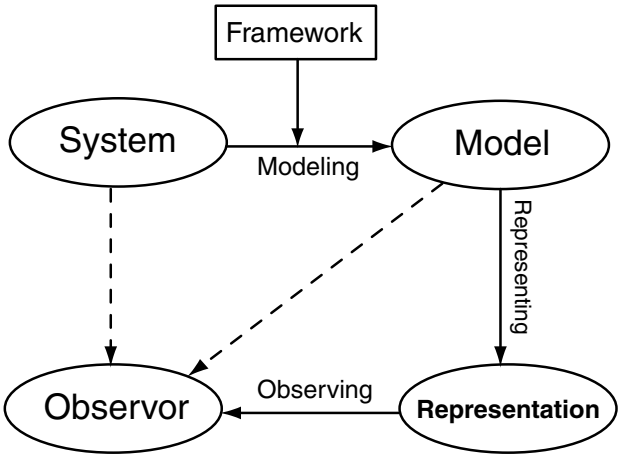

Figure 1. THE MODELLING OF A SYSTEM.

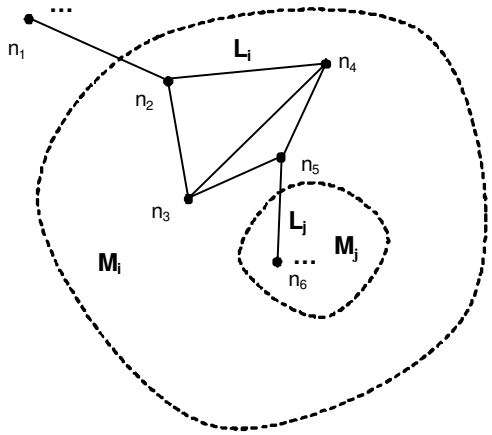

Figure 2. ABSTRACT GRAPH REPRESENTATION OF STRUCTURES.

3.1 Names and Links

Links can be represented as messages containing codes for two nodes and attributes. For instance, link L_i in Figure 2 can be represented as "$n_2 n_4 <$ attribute values $>$". The nodes may be those inside submodules, *e.g.*, link L_j in Figure 2 can be represented as "$n_5 M_j n_6 <$ attribute values $>$". Messages for attributes will be discussed in Section 3.2. This section will discuss encoding for node names. Now consider a module, and suppose that there are $N^{(n)}$ nodes and $N^{(m)}$ submodules. And 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.

A link can be represented as
<Node p Name><Node q Name>,
and the corresponding message length for name encoding is $L_p^{(n)} + L_q^{(n)}$
<Node p Name><Submodule t Name><Interface v Name in Submodule t >,
and the corresponding message length for name encoding is $L_p^{(n)} + L_t^{(m)} + L_{tv}^{(o)}$
<Submodule s Name><Interface u Name in Submodule s ><Node q Name>,
and the corresponding message length for name encoding is $L_s^{(m)} + L_{su}^{(o)} + L_q^{(n)}$
<Submodule s Name><Interface u Name in Submodule s ><Submodule t Name><Interface v Name in Submodule t >,
and the corresponding message length for name encoding is $L_t^m + L_{tv}^{(o)} + L_s^{(m)} + L_{su}^{(o)}$

and the message length for names of links in the module is

$$\begin{aligned} I^{(n)} &= \text{Sum of the lengths of all links in the module} \\ &= \sum_{j=1}^{N^{(n)}} L_j^{(n)} \times N_j^{(n)} + \sum_{j=1}^{N^{(m)}} L_j^{(m)} \times N_j^{(m)} \\ &\quad + \sum_{j=1}^{N^{(m)}} \left(\sum_{k=1}^{N_j^{(o)}} L_{jk}^{(o)} \times N_{jk}^{(o)} \right) \end{aligned} \tag{2}$$

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 encoding length for nodes, submodules and interfaces in submodules. Within the module, nodes and submodules are at the same level, so it is applicable to encode them with the same name table. For convenience, let "unit" be "node" or "submodule". If one unit j has N_j, which could be $N_j^{(n)}$ or $N_j^{(m)}$, connections to other units inside the same module, then encoding for this unit will be used N_j times in the total coding used to describe links inside the module. Let the frequency of unit j be p_j. According to Shannon's theory [21], the length of the label used to represent unit j should minimally be

$$-\log(p_j)$$

Then, total encoding length related to unit labels is:

$$-\sum_{j=1}^{N_i^{(u)}} N_j \log(p_j) \tag{3}$$

where $N_i^{(u)}$ is the number of units inside the module. p_j can be estimated as $N_j / \sum_{j=1}^{N_i^{(u)}} N_j$, and above formula is minimized. So Equation (3) can be rewritten as:

$$-\sum_{j=1}^{N_i^{(u)}} N_j \log(N_j / \sum_{j=1}^{N_i^{(u)}} N_j) \tag{4}$$

That is, the first two parts in Equation (2) can be estimated by Equation (4).

Now messages for interfaces must be considered. Since interfaces are always associated with some specific module, it is only necessary to encode interfaces so that interfaces inside the same module can be identified. In submodule j of the module, let interface k have been linked to $N_{jk}^{(o)}$ nodes outside the submodule. By the same process shown above but with only considering connections to other modules, $L_{jk}^{(o)}$ can be estimated as

$$L_{jk}^{(o)} = -\log(N_{jk}^{(o)} / \sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)}) \tag{5}$$

Therefore, the third term in Equation (2) can be estimated as

$$-\sum_{j=1}^{N^{(m)}} (\sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)} \log(N_{jk}^{(o)} / \sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)})) \tag{6}$$

and the total length of messages needed to encode names in links at level n is

$$I_n^{(n)} = \sum_{\text{all modules at level } n} I^{(n)} \tag{7}$$

$$= -\sum_{\text{all}} \sum_{j=1}^{N_i^{(u)}} N_j \log(N_j / \sum_{j=1}^{N_i^{(u)}} N_j) - \sum_{\text{all}} \sum_{j=1}^{N^{(m)}} (\sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)} \log(N_{jk}^{(o)} / \sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)})) \tag{8}$$

3.2 Attributes

It is feasible to separate the attributes associated with links into two categories: unordered attributes and ordered attributes. There is no order between different values of unordered attributes. Generally they 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 multi-state attributes and ordered attributes are considered.

3.2.1 Multi-state Attribute In a module, there may be N_T multi-state attributes associated with links, and for every multi-state attribute T_j, there are N_{T_j} different possible states. 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

$$p_{t_j} = \frac{N_{t_j}}{\sum_{t_j} N_{t_j}} \tag{9}$$

Then the message length used to encode multi-state attributes at in the module is

$$I^{(T_i)} = -\sum_{j=1}^{N_T} \sum_{t_j} N_{t_j} \log(p_{t_j}) \tag{10}$$

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

$$I_n^{(mA)} = -\sum_{\text{all modules at level } n} \sum_{j=1}^{N_T} \sum_{t_j} N_{t_j} \log(P_{t_j}) \tag{11}$$

3.2.2 Ordered Attributes Unlike multi-state attributes, usually there is precision associated with ordered attributes, especially continuous ones. In engineering, there is always 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 $(M_i = \lfloor A_i/\delta_i + 0.5 \rfloor) \rightarrow A_i$, then the message length for integer A_i could be $L_{A_i} = \log(M_i) + 2\log(\log(M_i))$ [15]. So the total message length used to encode those ordered attributes at level n is

$$I_n^{oA} = \sum_{\text{all attributes at level } n} L_{A_i} \tag{12}$$

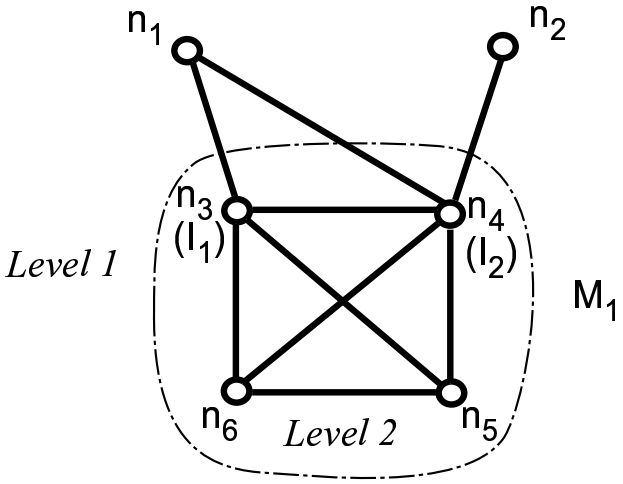

Figure 3. A SIMPLE EXAMPLE FOR INFORMATION MEASURE FOR MODULARITY OF GRAPH STRUCTURES.

3.3 Total Message Length

If linear aggregation is used, the total message length I:

$$I = \sum_n C(n)(I_n^{(n)} + I_n^{mA} + I_n^{oA}) \tag{13}$$

3.4 Example

One simple example is shown in Figure 3. At level 1, there are three units, *i.e.*, one module M_1 and two nodes n_1 and n_2, and at level 2, there are four nodes, n_3, n_4, n_5, n_6, inside module M_1 and two interfaces I_1, I_2. First, consider the message length at level 2. The code lengths for nodes and interfaces are shown respectively in Table 1 and Table 2. There are 6 links inside M_1 and they have all length $2+2=4$, so the total message length at this level $I^{(2)}$ is $6\times 4=24$. Now consider the message at level 1. The code lengths for nodes and links are shown respectively in Table 3 and Table 4. The total message length at this level $I^{(1)}$ is 14.90. With linear aggregation, the total information of the structure is $I^{(1)}\times C(1)+I^{(2)}\times C(2)$, where $C(1),C(2)$ are aggregation coefficients.

Table 1. CODE LENGTHS FOR UNITS IN M_1.

Node	# of Links	Code Length
n_3	3	$-\log 4=2$
n_4	3	2
n_5	3	2
n_6	3	2

Table 2. CODE LENGTHS FOR INTERFACES IN M_1.

Interfaces	# of Links	Code Length
I_1	1	$-\log 1/3=\log 3$
I_2	2	$-\log 2/3=\log 1.5$

Table 3. CODE LENGTHS FOR UNITS AT LEVEL 1.

Units	# of Links	Code Length
n_1	2	$-\log 2/6=\log 3$
n_2	1	$-\log 1/6=\log 6$
M_1	3	$-\log 3/6=1$

Table 4. CODE LENGTHS FOR LINKS AT LEVEL 1.

Link	Representation	Code Length
1	$n_1M_1I_1$	$\log 3+1+\log 3$
2	$n_1M_1I_2$	$\log 3+1+\log 1.5$
3	$n_2M_1I_2$	$\log 6+1+\log 1.5$
	Total:	14.90

4 VERIFICATION

In order to verify the information measure of graph structures, this approach should be compared to other existing measures or test it on some benchmark problems. Unfortunately, there is not much published work on measuring or quantifying measures of modularity, or benchmark problems. It does not appear feasible to test the measures in this way. However, this approach should at least be consistent with engineering intuition. To begin, consider a graph structure, and apply different hierarchial partitions (modular structures) to the structure. The information measure can be calculated for any partition, then the partition with the shortest message length is the best modular structure. This result will be compared with engineering intuition. For the first step, the graph structure will be partitioned into 2 levels.

Given a graph structure, this process of partitioning the graph into different hierarchical clusters involves searches in a large space with a complicated landscape. Garey [7] has shown that a k-way graph partitioning problem, which splits a weighted

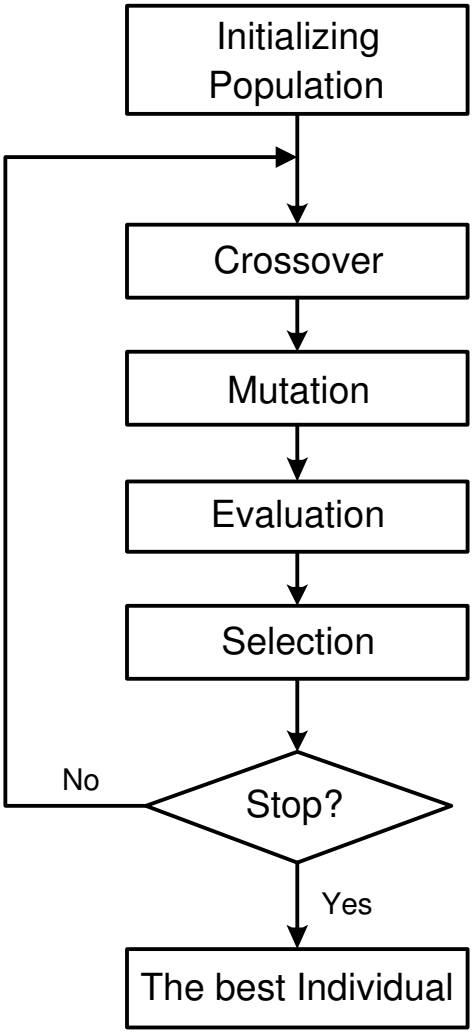

Figure 4. GENERAL STRUCTURE OF GENETIC ALGORITHM.

undirected graph into k clusters, is NP-complete. One suitable way to search such difficult space is genetic algorithms, search procedures based on the mechanism of natural selection.

4.1 Algorithm

Genetic algorithms were introduced in [11], and they were subsequently developed by Goldberg *et. al.*, [9]. A general structure of genetic algorithms is shown in Figure 4 in flowchart form.

In order to use genetic algorithms, We should provide

1. *Encoding schemes*. Since GAs work with a coding of the parameter set, not on parameters themselves, we need some encoding scheme. The scheme used in the following experiments will be discussed in Section 4.1.1.
2. *Ways to initialize population*. A common way is to randomly and uniformly sample the search space, which is used in the following experiment.
3. *Genetic operators*, which include mutation and crossover operators. The operators in this experiment will be discussed further in Section 4.1.2.
4. *Evaluation Function*, which is used to tell how fit individuals are as solutions to the problem. Since here we are trying to find the best modular structures of abstract graphs, it is obvious to use information measure of modular structures as indicators of fitness, and define the fitness of genomes(x) as:

$$\text{The Fitness } f(x) = \frac{N}{I(x)} \tag{14}$$

where $I(x)$ is information measure of the modular structure encoded as genome x, 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 how individuals are selected from this generation and their offsprings to form next generation. There are many different selection mechanisms such as rank-based, roulette wheel selection, tournament selection, and deterministic selection. Here roulette wheel selection is used to selection parents, and deterministic selection for generational selection. One important parameter related to the second kind of selection is 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 algorithm have a low generation gap. In these experiments, steady state genetic algorithms are used.

4.1.1 Encoding Scheme

As shown above, partitioning is used to cluster the vertices in the graph. The only thing needed to be encoded is which cluster a vertex belongs to. It's unnecessary to encode links. The genome structure is shown in Figure 5. Two arrays are used to represent a modular structure, one array for vertex indices and one array for tag indices used to separate vertices into different clusters. For the example for the genome shown in Figure 5(b), the modular structure is $\{1,2,3\},\{4,5,6,7,8\}$ and $\{9,10,11,12\}$.

4.1.2 Genetic Operators

1. *Crossover.* Given two parents, the operator compares them to find two clusters with most common vertices and exchange them, and get rid of the repeated vertex indices in other clusters and randomly merge absent vertex indices into the cluster which is neighbor to the index's original cluster, and at the same time make changes to tag arrays. For example, two selected parent genomes are one with vertex array

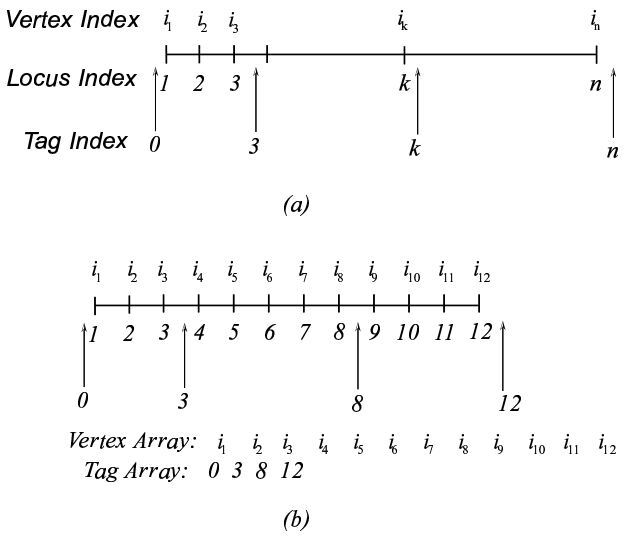

Figure 5. GENOME STRUCTURE. (a) GENERAL STRUCTURE, (b) AN EXAMPLE.

$[1,5,8\ \|,3,9,2,6,4\ \|,11,10,7,12]$ & tag array$[0,3,8,12]$ and one with vertex array $[5,8,11,2\ \|,3,9,4,6\ \|,1,10\ \|\ ,12,7]$ & Tag array $[0,4,8,10,12]$. The most common clusters are $\{3,9,2,6,4\}$ and $\{3,9,4,6\}$. We exchange them, and delete or merge other repeated vertices in other clusters and get two offsprings: one with vertex array $[1,5,8,2,\|\ 3,9,4,6\ \|,11,10,7,12]$ & tag array$[0,4,8,12]$ and one with vertex array $[5,8,11\ \|,3,9,2,6,4\ \|,1,10\ \|,12,7]$ & tag array$[0,3,8,10,12]$.

2. *Mutation.* There are two kinds of mutation operators: one on tag arrays and the other one on vertex arrays.

   Tag Mutation. Deleting and adding a tag in arrays; or drift tag position between its two neighbor tags.
   Vertex Mutation. Moving randomly chosen vertex from the cluster it belongs to to another randomly chosen cluster; Reordering the vertex index order in vertex arrays. One reordering operator is inversion. For instance, a vertex array $V[2,5,3,1,6]$ will become $V[6,1,3,5,2]$ after inversion.

4.2 An Example

A graph with 16 vertices is shown in Figure 6. In genetic algorithms, population size is set 50, crossover rate at 0.9, mutation rate at 0.2, and the aggregation function $C(n)$ is set as $C(n) = 0.5 * (n_{\max} - n) + 1$. Figure 7 shows the convergence of genetic algorithm, and Figure 8 shows the best modular structure found by the algorithm. In another experiment, $C(n)$ is set as $C(n) = 2.0 * (n_{\max} - n) + 1$, the result structure is shown in Figure 9. It shows that the aggregation should not be too strong. In fact the aggregation function is to balance the significance of the

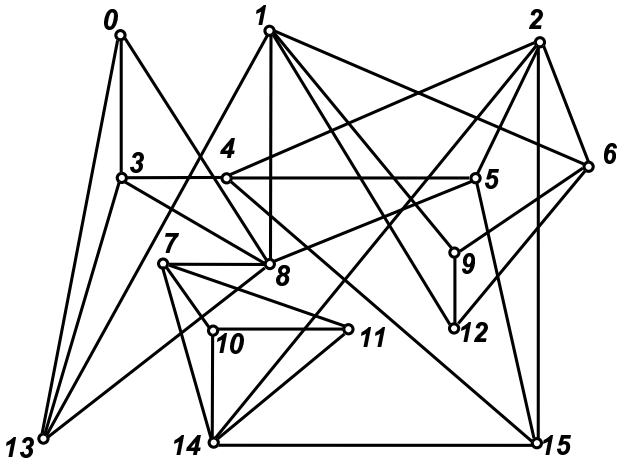

Figure 6. A GRAPH.

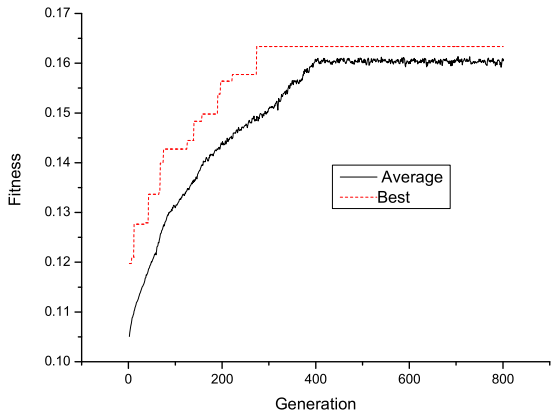

Figure 7. THE CONVERGENCE OF GENETIC ALGORITHM.

modularity at different levels. If the aggregation emphasizes the lower levels too much, the measure will favor the structures with less modules at lower levels. As shown in the Figure 9, there are only two modules.

5 DISCUSSION AND FUTURE WORK

Here, the primary result on an example abstract graph shows that the information measure of modularity could be applicable. However, the encoding methods for attributes and the measure on hierarchical modular structures still need to be verified further on more complex and real physical system. Some further work is needed to study the effect of aggregation functions, and to find the best aggregation function (and coefficients) for engineering practice.

The evolutionary technique showed in Section 4 could be

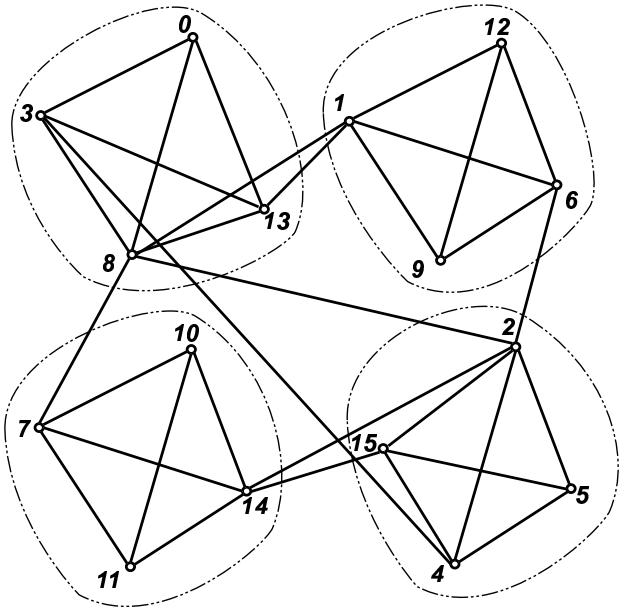

Figure 8. THE BEST MODULAR STRUCTURE FOUND FOR THE GRAPH.

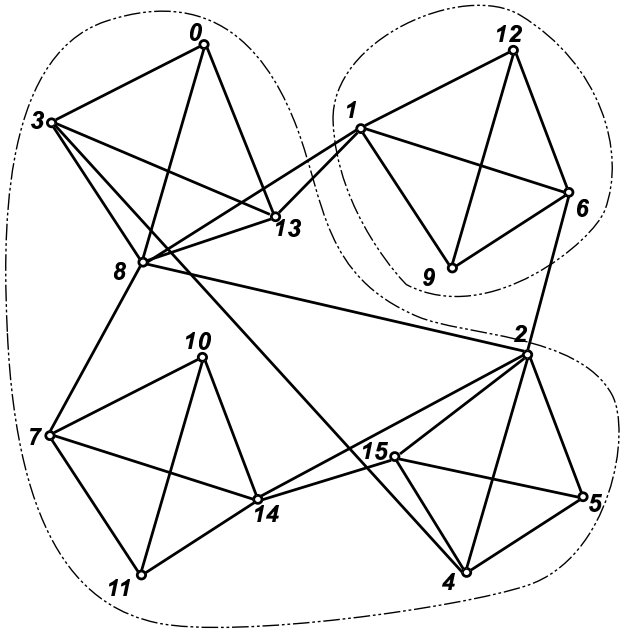

Figure 9. THE STRUCTURE FOUND WITH TOO STRONG AGGREGATION.

useful in engineering design to break a system up into modules. Additional methods to represent genomes operators need to be developed to apply this approach to more complex and hierarchial modular structures.

ACKNOWLEDGMENT

REFERENCES

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

[2] BARRON, A. R., RISSANEN, J., AND B.YU. The minimum description length principle in coding and modeling. *IEEE Trans. Information Theory 44* (1998), 2743–2760.

[3] BENNETT, C. H. *Logical Depth and Physical Complexity*. Oxford University Press, Oxford, 1988, pp. 227–257.

[4] CHAITIN, G. J. Algorithmic information theory. *IBM Journal of Research and Development 21* (1977), 350.

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

[6] FERNANDEZ. Integration analysis of product architecture to support effective team co-location. Master's thesis, MIT, 1998.

[7] GAREY, M., AND JOHNSON, D. *Computers and intractability: A guide to the theory of NP-completeness*. SFreeman and Company, 1979.

[8] GELL-MANN, M., AND LLOYD, S. Information measures, effective complexity, and total information. *Complexity 2*, 1 (1996), 44–52.

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

[10] HANSEN, M. H., AND YU, B. Model selection and the principle of minimum description length. *J. American Statistical Association 96* (2001), 746–774.

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

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

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

[14] KOLMOGOROV, A. N. Three approaches to the quantitative definition of information. *Problems of Information Transmission 1* (1965), 1–11.

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

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

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

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

[19] RUECKL, J. G., CAVE, K. R., AND KOSSLYN, S. M. Why

are “what” and “where” processed by separate cortical visual systems? a computational investigation. *Journal of Cognitive Neuroscience 1* (1989), 171–186.

[20] SCHILLING, M. Towards a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review 25*, 2 (2000), 312–334.

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

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

[23] SIMON, H. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[24] SOLOMONOFF, R. J. A formal theory of inductive inference part I. *Information and Control 7* (1964), 1–22.

[25] STARR, M. K. Modular production - a new concept. *Harvard Business Review 43*, 6 (1965), 131–142.

[26] TRAUB, J. F., AND WOZNLAKOWSKI, H. Perspectives on information-based complexity. *Bulletin of the American Mathematical Society 26*, 1 (1992), 29–52.

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

[28] WAGNER, G. P., MESEY, J., AND CALLABRETTA, R. Natural selection and the origin of modules. In *Modularity: Understanding the Development and Evolution of Complex Systems*, W. Callebaut and D. Raskin-Gutman, Eds. MIT Press, Cambridge, MA, 2002.

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

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

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

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