Information-Theoretic Methods for Modularity in Engineering Design

Thesis by

Bingwen Wang

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2007

(Defended August 9, 2006)

© 2007
Bingwen Wang
All Rights Reserved

Acknowledgements

First of all, I am very grateful to my advisor, Professor Erik K. Antonsson, for his guidance, encouragement, full support, and patience. He always gave me much freedom to explore what I am interested in and learn the things I enjoy. This made my study at Caltech a pleasant experience. Without his support and advice, I wouldn't have chance to explore information theory and therefore this thesis would be impossible.

Thanks to Professor Robert J. McEliece, who inspired me in the field of information and coding theory. He always has new ideas and innovative ways to approach different subjects. I have learned much from his inspiring ideas, expertise and research attitude. Also I would like to thank my other committee members–Professor Joel W. Burdick, and Professor Ken Pickar for their support and helpful discussions.

I owe my special thanks to my wife, my brother, my parents, and all my friends for their love and support.

Abstract

Due to their many advantages, modular structures commonly exist in artificial and natural systems, and the concept of modular product design has recently received extensive attention from the engineering research community. Although some work has been done on modularity, most of it is qualitative and exploratory in nature, and little is quantitative. One reason for this gap is the lack of a clear definition of modularity. This thesis begins with a detailed discussion on the concepts of "modularity" and "module".

Based on the background presented here, a mutual information-based method is proposed to quantify modularity. The method is based on the view that coupling is information flow instead of real physical interactions. Information flow can be quantified by mutual information, which is based on randomness (or uncertainty). Since most engineering products can be modeled as stochastic systems and therefore have randomness, the mutual information-based method can be applied in very general cases, and it is shown that the commonly existing linkage counting modularity measure is a special case of the mutual information-based modularity measure.

The mutual information-based method is applicable to final design products. But at the early stage of the engineering design process, there are generally only function diagrams. To exploit the benefits of modularity as early as possible, a minimal description length principle-based modularity measure is proposed to determine the modularity of graph structures, which can represent function diagrams. The method is used as criteria to hierarchically decompose abstract graph structures and the real function structure of an HP printer by evolutionary computation. Due to the specialty of genome representations in evolutionary computation, new genetic operators are developed to determine optimal hierarchical decompositions.

This quantitative modularity measure has been developed to synthesize modular engineering products, especially by evolutionary design. There are many factors affecting evolving modular structures, such as genome representation, fitness function, learning, and task structure. The thesis preliminarily studies the effects of the modularity of tasks on the modularity of products in evolutionary computation. Using feed-forward neural networks as examples, the results show that the effects are task-dependent and rely on the amount of resources available for the tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Ubiquity of Modularity

Modularity is all around us:

- In Figure 1.1, you can see an Indian looking towards you or an Eskimo with his back to you looking into a cave, but you can't see them both at the same time because our minds have "temporal *modularity*" [76]. That is, different computational configurations of nerve systems cannot exist contemporaneously.

- Neuroscientific evidence [31, 66] shows that there are two distinct cortical pathways in human brains for "where" and "what" tasks. Nervous systems have a ventral (temporal) pathway for recognizing the identity of an object and a dorsal (parietal) pathway for identifying its location.

- In biological evolution processes, it took about three billion years for nature to evolve single-celled organisms to multi-celled organisms while it only took about five million years to evolve from multi-celled organisms to now existing mammals. Why did it take so long for single-celled organisms to evolve into multi-celled organisms? One reason appears to be that *modularity* hastens the evolution process. *Modular* structures can lead to rapid adaptation to environments because by adding, subtracting, or modifying submodules, incremental changes can be more quickly tried and either adopted or rejected.

- There are a small number of kinds of single-celled organisms, while there are thousands of billions of organisms existing now in the earth. This is also due to their *modular* structures. *Modular* structures can exponentially increase variety.

- Some aspects of language processing also appear to be *modular*. Recent research [51] shows that competence in learning a language appears to be quite independent of general problem-solving skills.

Figure 1.1: An ambiguous image: Eskimo or Indian.

- Social structures and organization structures in human activities are *modular*. For example, the *modular* organization structure of the Executive Office of the President of the is shown in Figure 1.2

- People adopt *modular* methods to tackle complex problems. This *modularity* is mainly due to people's limited short-term memory, which is key to solving many intellectual problems. Because of this limitation, people can only handle one relatively small and easy problem at a time. This makes people decompose a complex problem into subproblems, therefore in a modular way.

- Many engineering designs are *modular*, probably due to people's modular way of tackling complex systems. A Boeing 747-400 has six million parts, and Pentium 4 has as many as 55 million transistors. As shown in Figure 1.3, they both are *modular*. It is their *modular* structures that make such complex systems manageable.

- Computer programming languages have evolved from structural languages like Basic, Fortran to C++ and object-oriented programming. It is this *modular* design methodology that makes it possible to develop large and complex applications.

- This thesis is also *modular*. First the structure is *modular*. It has chapters, and each chapter has sections, which contain subsections. Second it's *modular* in semantic sense. Each chapter has relatively independent contents.

  ...

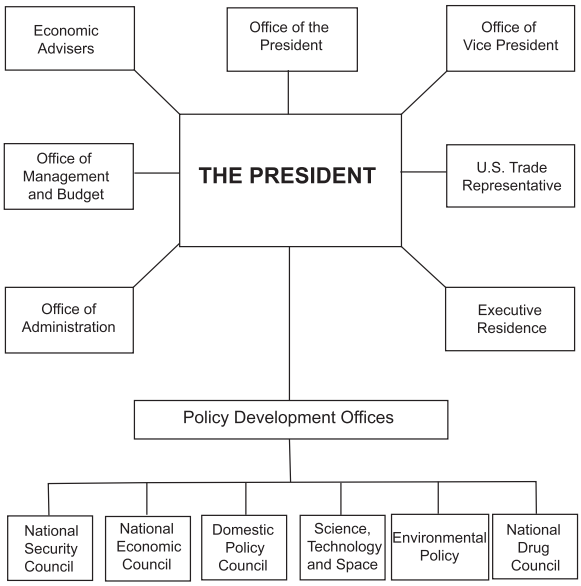


Figure 1.2: Executive offices of the president of the U.S.

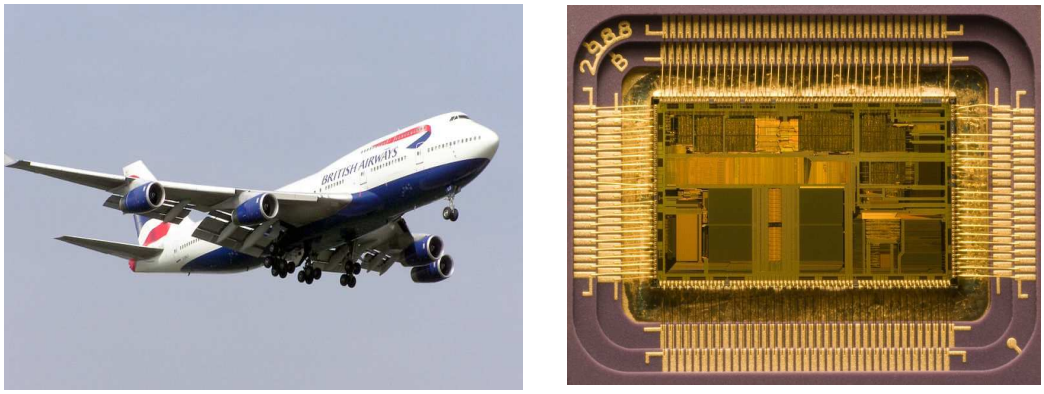


Figure 1.3: Modular structures in airplane and CPU.

- This list is *modular* with a collection of independent facts and observations organized into bulleted points.

The commonality of all of the elements in the above list is their *modularity*. As listed above, modular structures are widely employed in artificial and natural systems, and modularity is a necessity for the development of highly complex systems and a critical issue in understanding large complex systems. The universality of modularity is mainly due to its many advantages such as expanding humankind's limited intelligence capabilities to manage complex systems by making large systems more manageable, reducing cost [85], increasing flexibility [3, 85, 100], and boosting the rate of innovation [3]. The benefit of modularity will be discussed in more detail in section 2.2. Due to its wide use, its importance for understanding complex systems, and many advantages, modularity has recently received a lot of attention in many different fields, such as engineering design [3, 98, 100], production and manufacturing [67, 92], industrial infrastructure [58, 86], artificial intelligence [11], neural computation [5, 12, 32, 36], software engineering [74, 73], robotics[7, 43], brain science [50, 82], and biological evolution [8, 13, 101, 102].

1.2 Motivation: The Ambiguity and Informality of Modularity

Though the terms "modular" or "modularity" are used throughout the engineering literature, little empirical or even theoretical work has been done in either measuring or quantifying the modularity of artificial or natural systems. One reason for this gap is the lack of a clear definition of modularity. Even though the literature offers some definitions for modularity in different fields from different views, there seems to be no unified view on such questions as what exactly modularity is and what determines a "module." So, it is necessary to clarify the concepts of "modularity" and "module" in order to take advantage of modularity to provide good guidelines for synthesizing modular structures or products.

Most literature relies on informal explications of the general concepts of "modularity" and "modular," which can be easily understood by people but not so easily by machines (programs) since people are intelligent enough to draw on their knowledge of systems. Therefore, an additional motivation for this work is to provide a technical framework to make the definitions of modularity and related terms formal, precise, and explicit for automatically evolving modular engineering designs. One specific computational motivation is to use quantitative measures of modularity as criteria to

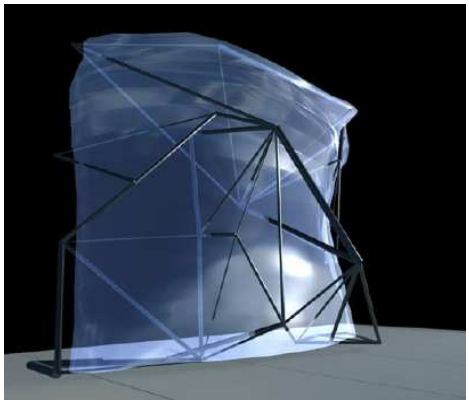


Figure 1.4: Non-modular truss structure synthesized automatically by MOSS [96]

synthesize modular structures, such as a truss. The example shown in Figure 1.4 is one automatically synthesized by MOSS, a computer program [96]. The structure is highly non-modular. It is necessary to propose a quantitative modularity measure to help synthesize modular structures.

1.3 Chapter Overview

In this work, the definition of modularity is clarified first, then information-theoretic measures for modularity are developed, and then are verified as criteria to decompose graph structures and function structures in engineering designs. At last, they are used to evolve modular artificial neural networks.

In the thesis, properties and definitions of "modularity" and "module" are qualitatively discussed in Chapter 2. A framework of mutual information-based measures of modularity is proposed in Chapter 3, and the framework can be used to quantify modularity of dynamic behaviors in stochastic systems, design processes, manufacturing and other processes. Minimal Description Length (MDL) principle-based measures for structural modularity of graph structures are discussed in Chapter 4. Chapter 5 uses the measures to decompose graph structures and function structures in engineering design, and Chapter 6 presents preliminary studies on how to evolve modular topology structures of artificial neural networks.

1.4 Contributions Made in the Thesis

The thesis contributions are listed below:

1. Several key characteristics of "modularity" and "module" are identified and discussed. Based on the discussion, a framework to quantify modularity is established. See Chapter 2 and Chapter 3.

2. Information flow is introduced as a quantification of interactions between systems. Based on the information theoretic view, a mutual information-based measure is proposed as a quantitative measure of modularity. The information-theoretic measure can be applied to more general systems than those real physical interaction-based methods, and it is shown that the existing linkage counting methods are a special case of the mutual information based method. See Chapter 3.

3. Since the mutual information-based method is limited to existing devices or systems, an MDL-based modularity measure is proposed for structure modularity of graph representations, which is common at the early phase of engineering design. See Chapter 4.

4. These information-theoretic methods are demonstrated by being used as criteria to hierarchically decompose abstract graphs and function structures. New genetic operators for tree representations of modular structures are developed in evolutionary computation, which is used to hierarchically decompose graph structures by evolutionary computation according to the information-theoretic measures. See Chapter 5.

5. The aim of developing quantitative modularity measures is to produce modular engineering products. To evolve modular structures, the first step is to understand the factors that will affect the synthesis of of modular structures by evolutionary design. The thesis preliminarily studies the effects of the modularity of tasks on the modularity of structures generated by evolutionary computation and shows that the effects are task and resource dependent. See Chapter 6.

1.5 Design as an Evolutionary Process

Engineering design is discussed from the perspective of biological evolution in this section. Comparing design processes to natural evolution will help to understand the benefits of modularity, origins of modules, formation of modular structures, and to develop modularity measures.

1.5.1 Design Space

The purpose of design is to synthesize some physical products to satisfy the customer requirements or customer attributes (CAs). However, design processes do not directly work on CAs , but instead operate on functional or performance requirements ($\bar{P}$), which are the set of requirements that completely characterize the design objectives based on customer attributes. The output of a design is a complete description of an artifact. The description in turn can be broken down into basic units, called *design parameters*. For example, to design a cylinder, the radius and height will be the design parameters, and there are probably some other design parameters such as material and

color. Let's denote the set of design parameters as $(x_1, x_2, \cdots, x_n)$, and suppose x_i takes some value from domain R_i. Domain R_i could be an interval in $\mathbb{R}$, such as the height of cylinder, or it also could be a discrete value from a set, such as the color. Then, the *design space* can be thought of as $R_1 \times R_2 \times \cdots \times R_n$. Any specific design can be viewed as a point in the design space.

1.5.2 Design Evolution

Biological evolution is a process where individuals with high fitness survive in a complex environment to pass their traits on to offspring. The traits of the offspring are determined by a mix of the parents genes (through crossover) and mutation. The evolution trajectory of an organism can be viewed as one in the evolution landscape, in which the altitude of the point equals the fitness of the organism. As an analogy to biological evolution, consider there is a landscape in design space. During design processes designers create variations of designs by changing design parameters, and move from one point to another in the landscape. Given adequate knowledge of determining where to move and the ability to move around in the landscape, designers could drive designs upward toward higher altitudes.

To formally describe design evolution, the following things are needed:

1. A design space which includes many designs (entities) and where designers can explore design alternatives.

2. A fitness measure, which the selection of modular design configurations is based on.

3. Sources of variation, which can explore many designs in the design space. A variation can happen to one design (entity) and produce a new one, like mutation operators in biological evolution, or it could happen to several entities and produce several new entities, like crossover operators which produce one or more new entities from two existing ones.

4. Mechanisms of selection, which include criteria to choose designs (entities) for the next generation (or iteration of the design process. In some contexts, selection mechanisms are two-valued functions. One value represents keeping a design (entity) for the next generation, and the other one means throwing away a design (entity).

5. Control units which control sources of variation and selection mechanisms. In typical design process, designers are the control unit, and their counterparts in biological evolution are complex environments affecting selection mechanism and mutation and reproduction processes controlling crossover and mutation.

The first two items above comprise of a *fitness landscape*, which can be formally defined as:

Definition 1 (Fitness Landscape) *A fitness landscape L is a triple $L = \langle D, R, f \rangle$, where D is the design space, $R \subset D \times D$ is a neighborhood relation on D, and $f : D \longrightarrow \mathbb{R}$ is a fitness (or objective) function.*

Peaks in a landscape are defined as:

Definition 2 (Peak) *A point $x \in D$ is a local peak if $f(y) \leq f(x), \forall y R x$ and a global peak if $f(y) \leq f(x), \forall y \in D$.*

With the definition of a fitness landscape, design evolution (process) can be defined as:

Definition 3 (Design Evolution) *A design evolution E is a quadruple $\langle L, V, S, C \rangle$, where*

- *L is a fitness landscape $\langle D, R, f \rangle$.*
- *V is a set of variation operators which are mappings from D^m to $D^n, m, n \in \mathbb{N}$.*
- *S is a selection mechanism, which is a mapping from D to $\{0, 1\}$.*
- *C represents a set of control rules.*

1.5.3 Difference between Design Evolution and Biological Evolution

Despite the strong parallels between biological and design evolution, the following differences are important to observe.

1. Moving around in the landscape in design evolution is not totally random since designers, the fundamental control units of design evolution, are capable of seeing and seeking values in design spaces while they are random in biological evolution. This is a key reason why organisms evolve much more slowly than engineering designs.

2. Biological selection pressures are on the whole systems, i.e., the combined genotypes and phenotypes.[1] Whereas in modular structures, design evolution can select for specific subsystems or modules.

3. The fitness landscape and evolution spaces can both vary with time in biological evolution and design evolution. Yet the speeds of changes in design evolution are much faster than those in biological evolution.

4. In design evolution, designers are the only control units, while in biological evolution, selection mechanisms and variations are probably controlled by different units. The variation sources in biological processes are reproduction processes which are controlled by some mechanism

[1] In the biology community, there is a debate on whether the biological selection is on genotypes, phenotypes, or both.

inside organisms, yet the selection is mainly controlled by the outside complex environment. Due to the separation of variation and selection in biological evolution, organisms can not choose their reproduction strategies according to the selection strategies of the outside complex environment. However, in design evolution, designers are the only control center. Therefore there is no strict separation of variation and selection. So, human designers can decide what variations to create and how to select by knowing both the selection and variation mechanisms.

5. In design evolution, human designers can develop models of their design processes and improve design processes. The relationship between variation and selection itself may be endogenous, adaptive, and the result of conscious design. Therefore, patterns of variation and selection in design processes are not "hardwired": they can and do vary across contexts and over time.

Chapter 2

Qualitative Modularity

2.1 Introduction

Due to their importance in understanding and managing complex systems, modular structures are favored in engineering practice. In order to produce modular structures, engineers need to understand the following questions: What is modularity? What is a module? And what characterizes modularity? Then a quantitative measure of modularity is needed. These are motivations for this thesis work.

Though there are already many qualitative discussion of modularity [3, 46, 61, 81, 85, 86, 100](refer to Section 2.4), they are not yet unified and there are still some very basic questions related to basic properties and quantitative measure of modularity that need to be clarified. This chapter mainly discusses those questions after a detailed discussion of the benefits and costs of modularity and a survey on existing views of the definition of modularity.

2.2 Benefits of Modularity

2.2.1 Product Evolution

In nature, single celled organisms dominated the Earth for billions of years. Multi-cellular organisms came into existence only about 500 million years ago. In mere tens of millions of years, they evolved so rapidly that they overtook three billion years of evolution of the single-celled organisms, replacing them as the dominant living things. The main reason for this is the modular structures of organisms. As discussed in Chapter 1, there are similarities between engineering design and biological evolution. Modular structures can also hasten product evolution and produce better performing designs. What is the real mechanisms behind these benefits? Based on the elements of the design evolution $E = \langle L, V, S, C \rangle$, modular structures can speed up the development process by:

- Smoothing the ruggedness of the evolution landscape: This can be explained by Kauffman's NK model. The model considers a system of N components, each of which in turn is in-

terconnected to $K(K < N)$ other components. The neighborhood relation R, i.e., the K connections for every component, can be chosen randomly. With the value of K varying from 0 to $N-1$, the system changes from totally decomposable to wholly integral. The configuration (or state) space X is $\{0,1\}^N$, and the fitness function is the average of the fitness values, f_i, of each of the N individual components. The fitness value, f_i, of component i is determined by the configurations (or states) of itself and its K neighborhood components $\{x_{i_1}, \cdots, x_{i_K}\}, \quad \{i_1, \cdots, i_K\} \subset \{1, 2, \cdots, i-1, i+1, \cdots, N\}$. That is,

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \cdots, x_{i_K}), \tag{2.1}$$

where f_i is a randomly generated mapping from $\{0,1\}^{K+1}$ to $(0,1)$. Then, the fitness landscape is $\langle X, R, f \rangle$. The results in Kauffman [53, 54] showed that

- For $K = 0$, there is only one peak in the fitness landscape.
- For $K = N-1$, the expected total number of local peaks is $\frac{2^N}{N+1}$.
- For K large, the average Hamming distance [1] between local peaks is approximately

$$\frac{N \log(K+1)}{2(K+1)}.$$

Those results imply that with N fixed, as K increases, i.e., the modularity decreases, the total number of local optima increases to a large number, and the landscape becomes increasingly rugged.

- Parallelizing the tasks or processes: The clear and well-defined interfaces of modules can enable design tasks to be decoupled, and every module has a well-defined function, therefore it is feasible to define a fitness function on the module, which makes design evolution possible on module levels. This decoupling results in the ability to complete tasks in parallel.

- Reducing design spaces: While decomposable structures can split the design space into several small ones and evolve on module levels, non-decomposable but not wholly integral systems can reduce the design spaces, though they can not split design spaces. Due to the reduced design space, modularity also boosts the rate of innovation [4], and leads to rapid trial-and-error learning [58].

- Making people (controllers) work more efficiently: People can only consciously work with a limited number of concepts at any time. Short-term memory, which is you used when solving an intellectual problem, only holds six or seven items [64]. This drives people to be specialized and solve problem in modular ways, and therefore they are more efficient in design.

[1] The Hamming distance of two binary strings is the number of different bits.

2.2.2 Production Variety

As to be discussed in Section 2.6, a module usually has well-defined interfaces and realizes some specific functions, and standard interfaces and well-defined functions of modules make it exchangeable. Usually there are several alternative components which can implement a functional element. The existence of alternative options makes it possible to construct a large variety of end products from a much smaller set of different components.

Simon [90] pointed out that "the more complex arise out of a combinatoric play upon the simpler. The larger and richer the collection of building blocks that is available for construction, the more elaborate are the structures that can be generated."

Suppose a system is composed of m modules and every module has $n_i, i = 1, \cdots, m$ alternatives, respectively. Then, by the multiplication principle, ideally there are $\prod_{i=1}^{m} n_i$ realizations. It is easy see that the number of combinations expands exponentially with respect to the number of modules. Though this exponential expansion would not happen in practice, modular structures can provide many options for production development. For example, IBM was able to create seven different printer motors by varying only face plates, gears, and end caps of the motor while leaving 25 other parts unchanged [100].

2.2.3 Cost Saving

The cost of designs includes development cost, test cost, and maintenance cost. All of those costs can be reduced by modularity. Firstly, a module has well-defined functions, and its interfaces are standardized so that its interactions with the rest of the product are minimized. This standardization allows the same component (module) to be used across product lines. This component sharing can dramatically reduce development costs.

Another cost saving factor is differential consumption [100]. Different parts of systems possibly have different consumption rates and require different materials, e.g., tires of automobiles. Modularity makes it possible to separate those parts from the rest of a system, and this strategy can drastically reduce the cost of the other "non-consumable" parts which otherwise have to be replaced with the same rate as consumable parts, and therefore, maintenance cost. Another maintenance cost saving factor is that maintenance can happen locally in modular structures in the sense that you can maintain modules separately.

Modular structures save test costs by reducing the possible test configurations. Let's consider the following two very simple scenarios, each dealing with testing a design with $2n$ test parameters, with each parameter having s states. This corresponds to s^{2n} possible combinations. Every test is assumed to have the same cost no matter what the size of the test, and all possible configuration combinations will be tested. Usually this number of tests will be very large, so in practice only a

fraction of all possible combinations are tested, and usually the number of tested cases is proportional to the total number. The following discussions on the extreme case will still provide some sense of test cost savings in modular structures.

One scenario is an integral design, and the other is decomposable design, which has two modules, with each having n test parameters. In the integral design, the system behavior is decided by the whole set of the test parameters, so the test can not be divided into different tests on subsets of the test parameters. Then, the possible number of tests is the total number of combinations s^{2n} .

In the decomposable design, since each module has a well-defined function, the test on the whole system can be divided into two tests on the two modules, respectively. Then, each has s^n possible test combinations, so there are $2s^n$ tests on module levels. How about integration tests on the system level? Let's assume each module has m output states, which is much smaller than the number of its possible state configurations, i.e., $m \ll s^n$. Then, the number of system level tests is m^2, so the total number of tests is $2s^n + m^2 \ll s^{2n}$.

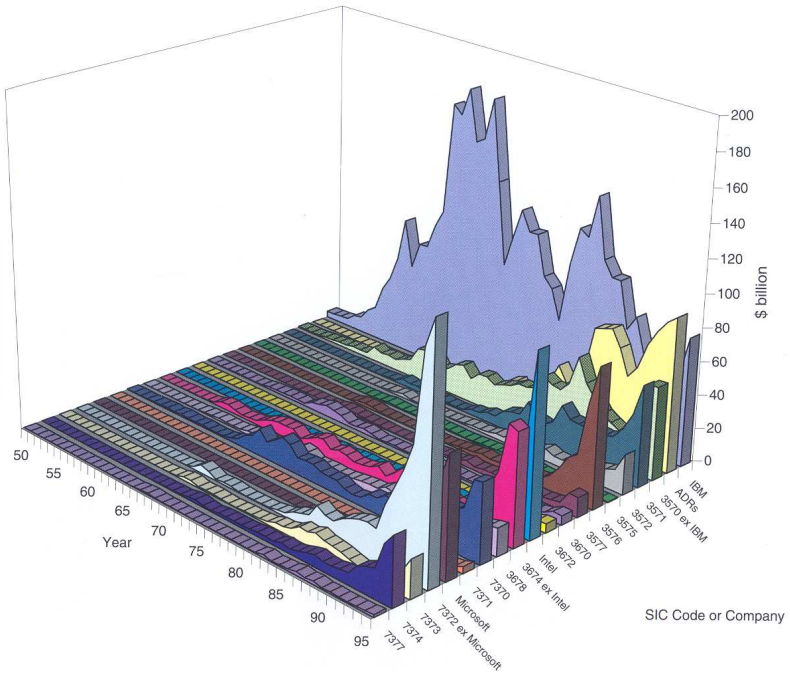
2.2.4 Specialization

Decoupling of production and design tasks of modular products enables individuals or firms to increase specialization [57]. Figure 2.1 shows the specialization in the computer industry due to the emergence of modular structures in computer architectures.

Specialization makes people concentrate their productive efforts on a rather limited range of tasks and encourages them to pursue specialized learning curves, increase their differentiation from competitors [86], and focus on a narrow area of knowledge, skills or activities. Specialization drives individuals or firms to have unusually effective or efficient performance of some particular function and increase their productivity and the reliability of products.

2.2.5 Reliability

As mentioned in Section 2.2.4, specialization caused by modularity can improve reliability. Additionally, in a fault tolerant system, modularity can further increase reliability with redundancy fixed. That is, modularity can further reduce redundancy with fixed reliability. Let's consider the following simple model. Suppose that there are n possible failure factors$\{f_1, \cdots, f_n\}$. For a non-modular structure A, the failure of any factor, f_i, can lead to the failure of the whole structure, and in order to keep the system running, the whole structure needs to be replaced. Yet, in an extreme modular structure B where n possible failure factors are separated into n modules, it is only necessary to replace the failed modules. Now consider the replacement probability, P. Assume all factors fail with the same probability p. Then:



Code	Category definition	Start date
3570	Computer and Office Equipment Except IBM	1960
3670	Electronic Components and Accessories	1960
3674	Semiconductors and Related Devices	1960
3577	Computer Peripheral Devices	1962
3678	Electronic Connectors	1965
7374	Computer Ternimals	1968
3571	Computer Processing, Data Preparation and Processing	1970
3575	Electronic Computers	1970
7373	Computer Integrated Systems Design	1970
3572	Computer Storage Devices	1971
7372	Prepackaged Software	1973
3576	Computer Communication Equipment	1974
3672	Printed Circuit Boards Computer Programming	1974
7370	Other Services Printed Circuit Boards	1974
7371	Computer Programming Services	1974
7377	Computer Leasing	1974

Figure 2.1: Market value of the U.S. computer industry [3]

.

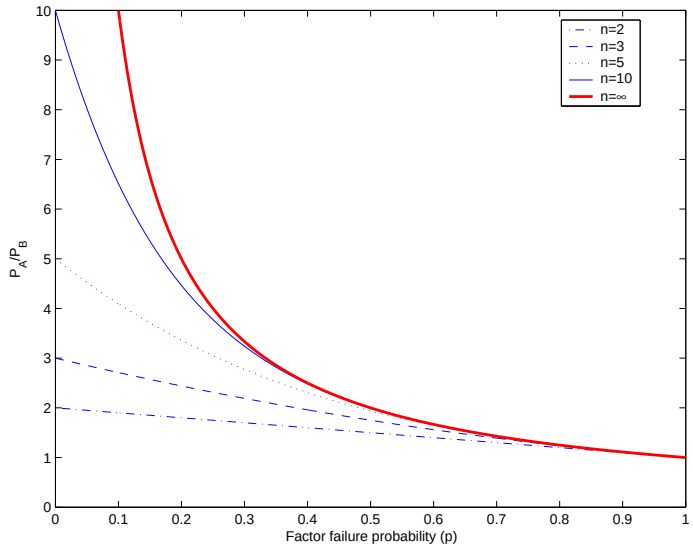


Figure 2.2: Modularity reduces redundancy.

- In the non-modular structure A,

$$P_A = 1 - (1-p)^n. \tag{2.2}$$

- In the modular structure B,

$$P_B = p. \tag{2.3}$$

Figure 2.2 shows the relation between P_A and P_B.

It can be seen from Figure 2.2 that the smaller p, the more benefits modular structures can gain. Usually in engineering, p is quite small, so P_A can be approximated by a Taylor expansion:

$$P_A = 1 - (1 - np + o(p)) = np + o(p) \approx np \tag{2.4}$$

And then

$$\frac{P_A}{P_B} \approx \frac{np}{p} = n \tag{2.5}$$

In another limit case: $n \to \infty$, then $P_A \to 1$, and therefore $\frac{P_B}{P_A} = \frac{1}{p}$.

This may partially explain why natural neural systems utilize a redundancy factor of 10,000 while current electronic systems use a factor of 2 to achieve very high reliability? The structures of neural systems are not so modular compared to modern electronic systems.

2.3 Cost of Modularity

Modularity does not come for free. Firstly, modularity is a result of a creation process. Along with increasing modularity, systems change from highly integral to modular structures and could be split

into subsystems (or modules). As shown in the evolution of computer architectures, before the highly modular system of today, the first general purpose computer "ENIAC" was highly interconnected and non-modular [4].

Performance is usually not optimal in modular structures. Given unlimited engineering resources, some performances of any particular product can be improved by reducing modularity. The performance improvement is often in terms of reduced size and mass. Modular designs can contain redundant physical structures and do not exploit as much function sharing [2] as possible.

Modularity may also increase some costs by introducing some redundancy and using standard components. When standard components are used in some particular applications while they are designed for more general applications, they may have excess capability.

Modularity can make products excessively similar by using standard components. Some auto makers have suffered from this problem because of the customer perception of too much sharing of styling and systems across models [100]. For this reason, much of the standardization of components that can be achieved with modularity is confined to hidden components, particularly in consumer goods [71].

2.4 Brief Survey on Definitions of Modularity

Due to the benefits discussed in the previous section, modularity became an important concept in engineering design. However, the concept of modularity and modular methods or structures are not new at all. The modular strategy originated very early. In the 1640s, in his *"A Discourse on Method: Meditations on first Philosophy,"* [17] Rene Descartes, the French philosopher and mathematician, proposed a modular method using decomposition as one of the basic ways to solve problems: *"... to divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution..."* The practice of modular strategies in engineering is also very early. In the 1900s, the Wright brothers used a modular design method to invent the airplane. They tackled the problem by solving three separate subproblems: lateral control, lift, and propulsion. To some extent, it was a modular design method that enabled their success [47]. The modular production paradigm in production is also very old. In 1914, Swan [95], an automotive engineer, extended the standardization idea across production and for large components, standardizing wheel sizes, hubs, bearings, axles, and fuel feeding mechanisms. Producing final products by assembling standard components dominated the early history of the U.S. auto industry through 1930.

In the 1960s, Alexander [1] and Simon [90] systematically studied methods to design complex systems and analyze their behaviors. In "The Architecture of Artificial," Simon [90] discussed a broad ranging set of systems, from business organizations to biological systems, that exhibit

[2] Function sharing is the implementation of several functional requirements by a single physical element in a design [99].

a property of being "nearly decomposable", and also argued that those complex systems often take a hierarchical form, whereby the system is composed "... of interrelated subsystems, each of the latter being in turn hierarchic in structure until we reach some lowest level of elementary subsystem." He described nearly-decomposable systems as those where "the short-run behavior of each of the component subsystems is approximately independent of the short-run behavior of the other components," and "in the long run the behavior of any one of the components depends in only an aggregate way on the behavior of the other components." Simon suggested that intra-module interactions are stronger than inter-module interactions in nearly decomposable (modular) systems. This view fits well with common intuitions and is widely adopted, but it has limits.

Firstly, only considering the strength of intra-module and inter-module interactions is sometimes overly simplistic when applied to complex dynamical systems. Sometimes the functional behavior of one module is strongly dependent on the inter-module interaction, despite a week interaction [109]. Secondly, it is necessary to consider the "size" of modules ("granularity"). The granularity of modules affects the intra-module interactions, therefore the ratio of intra-module and inter-module interactions.

Considering the effects of the granularity of modules, Wagner [102, 103] introduced function to limit the size of modules, although this view is limited to the field of biology. He provided three criteria for recognizing modular phenotypic units: "(1) collectively serve a primary functional role; (2) are tightly integrated by strong pleiotropic effects of genetic variation; and (3) are relatively independent from other such units."

Wagner's view is not the only one which associates modularity with functionality, even in the field of biology. Based on Fodor's work [21], Elman [18] stated: "A module [of mind] is a specialized, encapsulated mental organ that has evolved to *handle* specific information types of particular relevance to the species."

In engineering design, Pahl [71] defined modular products as "machines, assemblies, and components that fulfil various overall functions through the combination of distinct building blocks or modules" and different types of modules to implement different types of technical functions including basic, auxiliary, special, and adaptive.

Ulrich [100] argued modularity depended on two factors – similarity between the physical and functional architectures of designs and minimization of incidental interactions between physical components. A modular architecture includes a one-to-one mapping from functional elements in the functional structure to the physical components of the product and specifies decoupled interfaces between components. An integral architecture includes a complex (non one-to-one) mapping from functional elements to physical components and/or coupled interfaces between components [98].

Functionality can only be well-defined for components or systems, which can be clearly separated from other systems. This means that those views of associating functionality to modularity are only

suitable for decomposable systems, but not for non-decomposable systems. There are no clear definitions for module with primary functional roles. Furthermore, functionality is not the only aspect of modularity. Modularity is multi-dimension and it also includes other aspects such as physical, temporal, life cycle [67], and design process. For example, modularity in the brain can be categorized into three different types: architectural modularity, functional modularity, and temporal modularity [10]

Another question is on the definition domain of modularity. Some views limit modularity to modular structures. Galsworth [24] viewed modularity as having standardized and interchangeable components: "modular design is a unit or group of standardized elements or parts that may be used within a number of different products." Baldwin and Clark [4, 3] stated "a modular system is composed of units (or modules) that are designed independently but still function as an integrated whole." Langlois [57] stated "modularity is about how parts are grouped together and about how groups of parts interact and communicate with one other." Some other scholars view modularity as a general concept for any system. For example, Shilling [86] viewed modularity as a matter of degree, and "a complex system can be modular at various degrees."

2.5 Modularity

The general principles or properties of modularity that have been discussed in the literature surveyed in the above section are summarized here:

1. Modularity is a systematic concept and is a global characteristic of a system.
2. Modularity is related to intra-module and inter-module coupling.
3. Modularity is hierarchical. A modular system can be separated into subsystems, and subsystems can also be separated into sub-subsystems, and so on.
4. Modularity is multi-dimensional. Modularity is related to many different aspects such as physical structures, function structures, design processes, and time.
5. Modules usually realize some functions.

The following important aspects of modularity have not been discussed in the literature and need to be clarified.

1. *System vs. Model:* It is not common to directly study a system, but instead a model of the system. So, what people actually talk about is a model of a system. What then is the relation between the modularity of a system and the modularity of a model of the system? Furthermore, can the modularity of a system be well-defined?

2. *Relativity of Interactions:* Modularity compares inter- and intra-module interactions of a system. Should a modularity measure compare the total quantity of interactions or the interaction "density"?

3. *Hierarchy:* Modularity is hierarchical. How then is the modularity of a whole system affected by the modularity of its subsystems and aggregated from them?

4. *Decomposition:* If a system is decomposed, there are no modules or subsystems. Therefore, "inter-module" or "intra-module" is not well defined. That is, what is the specific meaning of "inter-module" or "intra-module"?

5. *Definition Domain:* Should modularity be limited to modular structures or be a universal concept in the sense that any system can be modular to some degree, and the definition of modularity should work across a broad range of cases?

6. *Quantifying Interactions:* What is the specific meaning of "couplings"? How to quantify interaction strength?

The first five questions are discussed in the details below, and the quantification of interactions (coupling) will be discussed in the next chapter.

2.5.1 System and Model

A system consists of elements and interactions among the elements. Modularity is used to describe the degree of interactions among the elements. So, modularity is related to the global behavior of a system, and this leads to a number of questions. For example, what should be considered "global behavior "? What constitutes the overall function of a system? Since the understanding of behaviors of a system depends on observers' knowledge, how does people's knowledge affect the modularity of the system?

Generally, systems, especially natural systems, are complex, and it is difficult or even impractical to understand behaviors of units in a system and interactions among them. This ignorance will give you the difference between the modularity you observe and the modularity the system really has. When you know a reasonable amount of information about functions and components in a system, you can reliably attribute modularity to the system. Otherwise, you may wrongly assign low modularity to a system which is really highly modular because of the ignorance. For example, before people really understood the behaviors of human's mind, the mind was perceived as a seamless, unitary system whose functions merged continuously into one another. With continuing studies, it was found that the mind has modular structures which consist of "a number of distinct, specialized, structurally idiosyncratic modules that communicate with other cognitive structures in only very

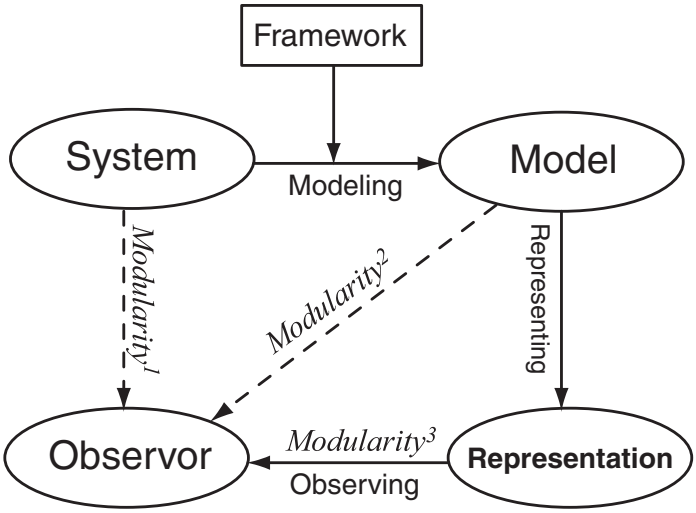


Figure 2.3: The modeling of a system. Modularity[1] is not well-defined. Modularity[2] = Modularity[3].

limited ways" [26]. The modularity of the mind is objectively there. However our answer changed from non-modular to modular.

In fact, there is no criterion to tell whether your knowledge of a system is complete or not. That is, it is really difficult to tell whether the modularity observed is the real modularity of the system or not. This does not mean, however that it is impractical to talk about modularity, although it is very difficult to directly measure the modularity of a system. Usually, a given system, especially a natural system, has too many contents, and it is impractical or unnecessary to arbitrarily accurately describe it, which is beyond our understanding and modeling abilities. So, the system should be approximately modeled under some framework and formalized by some representation languages, as shown in Figure 2.3. Now the behaviors of models and therefore modularity can be studied. Does this mean the measure of modularity will become subjective? If observers model a system and do activities related to modularity within an established framework, the modularity of the system can be determined with respect to the framework, and different observers' views on modularity should be consistent.

Then, it seems reasonable to define the modularity of a system as the modularity of the "best" model of the system, which is the "best" approximation of the system under the agreed framework. The problem is how to define "best" and the existence of the "best" model. Firstly, knowledge of a complex system usually increases along time, so the set of models of the system $\{Mod_i\}$ evolves. Secondly, it needs an ordering on the set of all possible models to define "best" formally. Let us denote the whole set of global behavior or information of a system as I. The behaviors or information described by a model of the system should be a subset of I, denoted as I_i, for Mod_i. Usually, the more information a model can describe a system, the better the model, so it is reasonable to assign the following ordering $\leq$ on the models: if $I_i \subset I_j$, then $Mod_i \leq Mod_j$. Then the question becomes

whether there is a maximal element in $\{Mod_i\}$ under the order $\leq$. The answer is negative since ordering $\leq$ is usually not a total one.[3] Suppose there are two model Mod_i and Mod_j describing information sets I_i and I_j respectively, where $I_i \neq I_j$. Unfortunately it's not guaranteed that there exists another model describing the information set $I_i \cup I_j$.

So, when people talk about modularity of a system, they are talking about modularity of a model of the system which is modeled under some default framework.

2.5.2 Multi-Dimensional

Modularity is multi-dimensional. Modularity of a specific aspect could be thought as the projection of the overall modularity. For example, biological modularity can be classified into three aspects: development, morphology, and evolution [13]. In engineering, modularity could include physical structure, function, temporal relationship, life cycles, etc. High modularity in one dimension does not necessarily mean high modularity in another dimension. A system could have strong inter-module dependencies in one aspect, though it may clearly be modular in other aspects. Here is a simple example.

Consider the simple Ising model [110] shown in Figure 2.4. There are 4 spins and connections between them. Each spin has a state 0 or 1, and the digital number over a line is the number of connections the line represents. The dynamic behavior follows the following update rules:

$$\begin{aligned} P(S_i(t+1)=1) &= \frac{1}{\sum_{j\neq i} c_{ij}} \sum_{j\neq i} c_{ij} S_i(t) \text{ and} \\ P(S_i(t+1)=0) &= 1 - P(S_i(t+1)=1), \end{aligned}$$

where c_{ij} is the number of connections between spin i and spin j, and $S_i(t)$ is the status of spin i at time t.

From the structure view, the system can be thought as a modular one, with spin 1 and 2 clustered as module M_1 and spin 3 and 4 clustered as module M_2. Does the modular structure imply modular dynamic behavior?

Let's denote the system state as a four-bit binary expansion $(S_1S_2S_3S_4)$. Given the initial spin state, the dynamic behavior of the Ising model is a Markov chain. From analysis on the transition probability matrix, there are two absorbing status (0000) and (1111), so the stationary state, S_f, of the dynamic system should be either (0000) or (1111). It is not difficult to get that the probability

[3] A total ordering $\leq$ on a set A is an ordering satisfying the following four properties:

1. $x \leq y, y \leq z \Rightarrow x \leq z$, Transitivity.
2. $x \leq y, y \leq x \Rightarrow x = y$, Anti-symmetry.
3. $x \leq x, \forall x \in A$, Reflexivity.
4. $\forall x, y \in A,$ either $x \leq y$ or $y \leq x$.

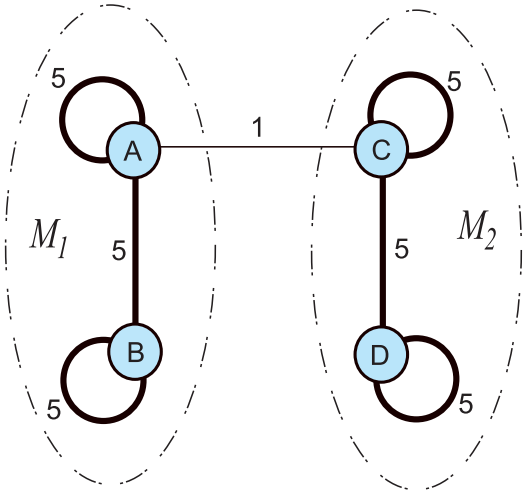


Figure 2.4: A simple Ising system.

of S_f is uniform, i.e., $P(S_f = (0000)) = P(S_f = (1111)) = 0.5$ if the probability distribution of initial states is uniform. No matter what the stationary state S_f is, the state of (S_1S_2), considered module M_1, can completely be inferred from the state of (S_3S_4), considered module M_2, and vice versa. This means that the dynamic behaviors of M_1 and M_2 are totally coupled.

2.5.3 Relativity of Interactions

While discussing the modularity of a system, it is necessary to compare the inter and intra-module interactions of the system. The weaker inter-module interactions and stronger inter-module interactions of a system, the better the modularity of the system. This means that modularity is related to the relative strength of inter-module interactions compared to that of intra-module interactions. This relativity makes *granularity*, i.e., the "size" of modules, come into consideration while defining modularity. "Size" could be the number of basic units or the amount of information from information-theoretic views. For example, there are two systems, S_1 and S_2, each having two subsystems. The two systems have the same "size" and inter-subsystem interaction strength, but different intra-subsystem interaction strength. S_1s subsystems are completely integrated, but S_2s are fully decomposable. If only based on the relative strength of inter-subsystem and intra-subsystem interactions, S_1 is more modular than S_2, but actually S_2 is more modular. In this case it is necessary to compare the modularity of subsystems, i.e., to decompose the subsystems further.

The larger the "size" of a module, the stronger the total intra-module couplings. Highly modular structures should favor strong intra-module interactions, and therefore a large size of module at a specific level. Consider the structure shown in Figure 2.5(a), where each subsystems has strong interactions inside, but the interactions between them are weak. If total interactions are considered, then the modular structure will be the one shown in Figure 2.5(b), not the usual and intuitive one

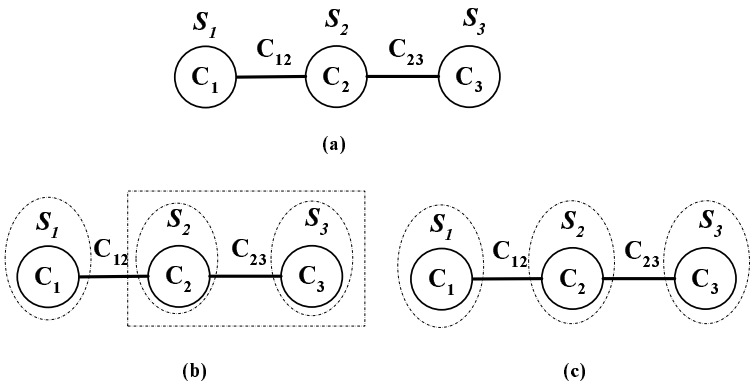


Figure 2.5: Relativity of interactions.

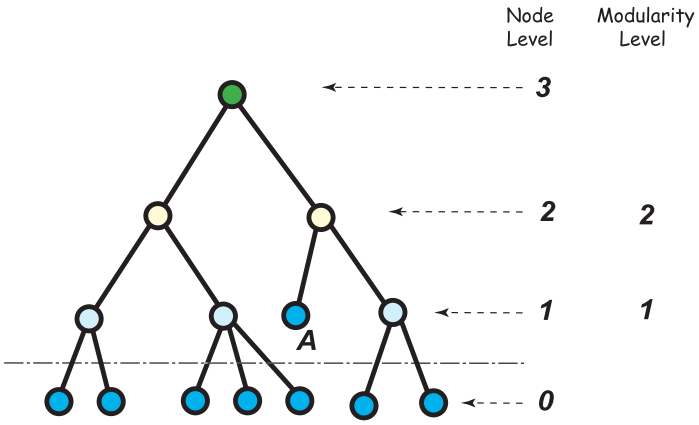


Figure 2.6: The levels of hierarchy.

which has three modules at one level, shown in Figure 2.5(c). However, if interaction "density," interactions normalized out by the "size"of module, is used, then the structure in Figure 2.5(c) will be better than the one shown in Figure 2.5(b).

So, to compare modularity, the interactions need to be normalized out by the "size" of modules, i.e., they become interaction "density".

2.5.4 Hierarchical

Hierarchical structures are common in complex systems. Complex systems take advantage of the hierarchy to regulate the states and structures of their subsystems, make them operate nearly independently of each other, and evolve rapidly and easily. The hierarchy looks like a tree. The levels of nodes are increasingly assigned from bottom to root, with the beginning at level 0, and the levels of modularity begin at node level 1, as shown in Figure 2.6.

Due to the hierarchical structure of a system, it is feasible to separately consider units in different

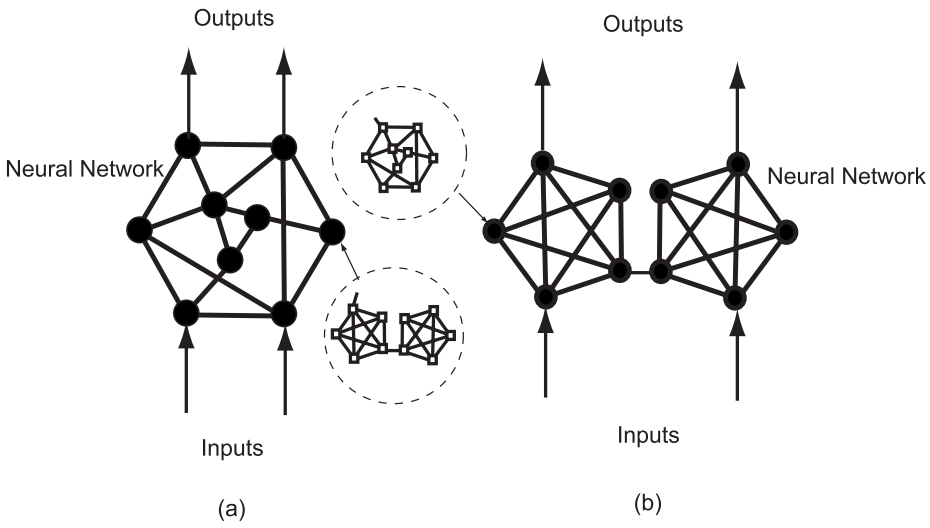


Figure 2.7: The effects of modularity at different levels on the overall modularity.

levels and define modularity for the units at a specific level. How then does the modularity of units at different levels affect the overall modularity of a model? Modularity of units at different levels of the hierarchy have different effects on the overall modularity. Modularity at a higher levels dominates those at lower levels, and it is not necessary that modularity of subsystems should be less than the overall modularity. For the two neural networks in Figure 2.7, units in the left one are highly integrated, but every unit has a modular structure, and whereas the right one is modular at higher level and the units at higher level have highly integral structures. It is easily told from general intuition that the right structure is more modular.[4]

One problem related to hierarchy is how to aggregate modularity at different levels. There are some constraints on the form of aggregation functions so that not all general aggregations [87, 69, 68] work here. Firstly, the modularity of subsystems at lower levels affects the overall modularity more than those at higher levels. This is consistent with practical intuition, in that the information/decisions at system levels are not less important than the information/decisions at module levels. This puts a special requirement on aggregation functions $f(M_1, \cdots, M_n)$:

$$\text{If } i > j, \text{ then } \frac{\partial f(M_1, \cdots, M_n)}{\partial M_i} \geq \frac{\partial f(M_1, \cdots, M_n)}{\partial M_j} > 0. \tag{2.6}$$

There are many different aggregation ways. One special case is linear aggregation. The overall modularity M is

$$M = \sum_{i=1}^{n} a_i M_i, \tag{2.7}$$

[4]This is in the sense of structure modularity. Modularity is multi-dimensional. As discussed later in this chapter, a system can have high modular structure while it has low modularity in other aspects.

and a_i should be an increasing function of i regardless of whether it is linear or nonlinear in i.

To get rid of the side-effects of the number of levels, a_is are required to satisfy

$$\sum_{i=1}^{n} a_i = 1. \tag{2.8}$$

The most common linear aggregation function is $a_i = Ci^{\alpha}$, where α is a positive real value, and C is a normalization coefficient such that equation 2.8 is satisfied. When $\alpha = 0$, it represents uniform aggregation.

Another important parameter related to hierarchical structures is the lower bound of sizes of subsystems– *resolution*, i.e., the lowest level a system can be decomposed into. Some systems, especially natural systems, can be divided into very small scales. For example, a computer is composed of a CPU, memory, motherboard, hard disk, and so on, all of which can be divided into smaller units. For instance, a CPU can have many different functional circuits, which can be decomposed into transistors and further decomposed into atoms, molecules, etc., and atoms or molecules consist of elementary particles at a higher level. Another interesting phenomena which requires the lower bound is self-similarity. In those systems, subsystems can iterate themselves inside themselves, and the iteration may be infinite. Usually, the resolution is decided in the framework under which the system is modeled. So, it is not necessary to discuss the resolution when modularity of a model is discussed.

2.5.5 Decomposition

While modularity is defined to compare interactions or couplings among different parts, components, or subsystems, how do "inter-module or intra-module" come to existence in a non-decomposed system since there are no modules or subsystems in such a system? In this case, an imaginary decomposition of the model is assumed to exist so that the couplings inside clusters (modules) and between clusters (modules) can make sense. Then, what is the criterion to partition a system into an imaginary decomposition? What is the measure of the "size" of a subsystem in the imaginary decomposition? And what kind of property of a set of elements allows them to be considered as a component/subsystem?

Many existing views of modularity associate functionality to modularity, and use function to decompose a system. For a fully modular design, there is a one-to-one correspondence between each functional element of the design and a single physical component [100]. As Henderson and Clark [39, 14] defined, a component is a physically distinct portion of the product that embodies a core design concept [14] and performs a well-defined function [39]. This is consistent with our intuition that a real component/subsystem should perform some functions.

However, functionality may not be a good criterion to decompose a system into a subsystem.

Firstly, there is no such thing as a function category which is complete, standard, and formal. Secondly, as discussed in Section 2.5.2, functionality is not the whole point of modularity. There are other kinds of modularity. Finally, it is even worse when those systems are not completely modular. In this case, there is no one-to-one mapping between function domain and physical domain, and there are no physical modules or subsystems, such as the neural network structure shown in Figure 2.9.

Modularity is multi-dimensional. Modularity in one dimension does not determine the modularity in another dimension, and any specific one among those modularities is not a sufficient and necessary condition for modularity of the system. And therefore any specific aspect can not become the partition criterion. Since there is no specific decomposition criterion for a system, all "possible" decompositions are considered. A decomposition clusters all elements of the system into units, denoted as U, which are defined as a collection of elements of a system.

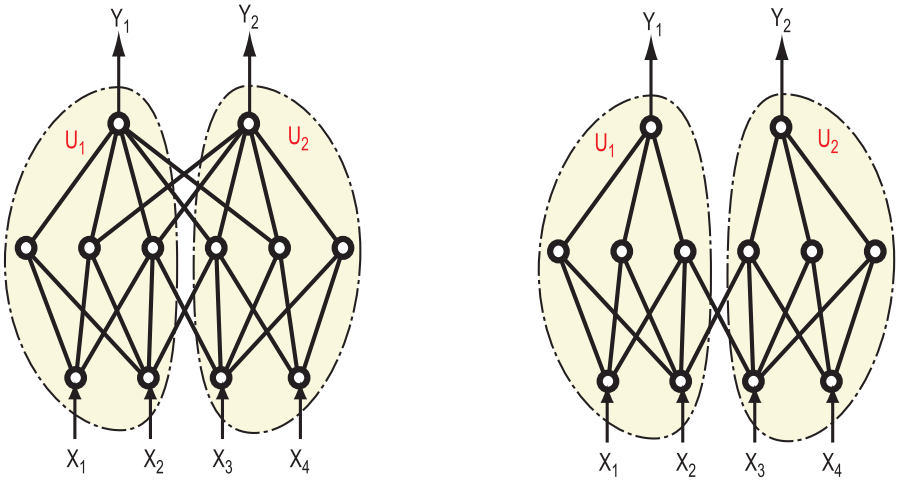
Consider different decompositions of two different networks, one in the left side of Figure 2.8 and the other one in right side of Figure 2.8. In the top decompositions, intuitively the right one is more modular than the left one. However, in the bottom decompositions, the left one is more modular than the right one since they have the same U_1 unit and interactions between U_1 and U_2, yet the left U_2 unit has stronger couplings inside than the right U_2 unit. These two different decompositions give different results on modularity, so it is necessary to compare the modularity of their maximal decompositions in order to compare the modularity of two systems.

Decompositions of a system are hierarchical. Therefore the units are organized as a hierarchy according to their relations with other units inside the model. The neural network in Figure 2.9(a), which calculates Y_1 and Y_2 from the four inputs I_1, I_2, I_3, and X_4, has two level 1 units, U_1 and U_2, respectively realizing output Y_1 and Y_2. U_1 has elements of $\{I_1, I_2, H_1, H_2, H_3, H_4, O_1\}$,[5] and U_2 has elements of $\{I_3, I_4, H_5, H_6, O_2\}$. U_1 can be decomposed further as two child units U_{11} and U_{12}. For example, unit U_{11} has elements $\{I_1, H_1, H_2\}$. The right tree in Figure 2.9 represents the hierarchical relation of the different units. The lowest module is called leave modules. Formally,

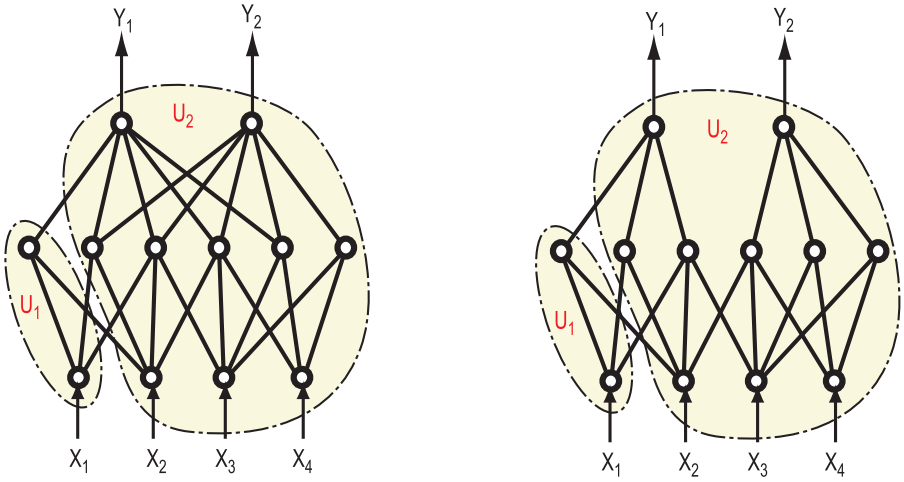
Definition 4 (Leave Module) *A module which has no submodules.*

As discussed in Section 2.5.3, the lower bound size of modules affect the modularity of a system. It is necessary to set up a threshold for decomposition. Threshold could be the number of elements or information inside subsystems. Also, it is possible that there are some constraints on the decomposition. For example, if the networks in Figure 2.9 are trained to learn two functions, one mapping (X_1, X_2) to Y_1 and the other one mapping (X_3, X_4) to Y_2, the possible decompositions should cluster nodes X_1, X_2, Y_1 to the same set. And, similarly, to (X_3, X_4, Y_2). Then, modularity of a system is taking maximum over the modularity of all the following feasible decompositions.

[5] In fact, it should include the connects and their weights between those nodes. For concision, we omit them in the following discussion.



(a)



(b)

Figure 2.8: Different decompositions.

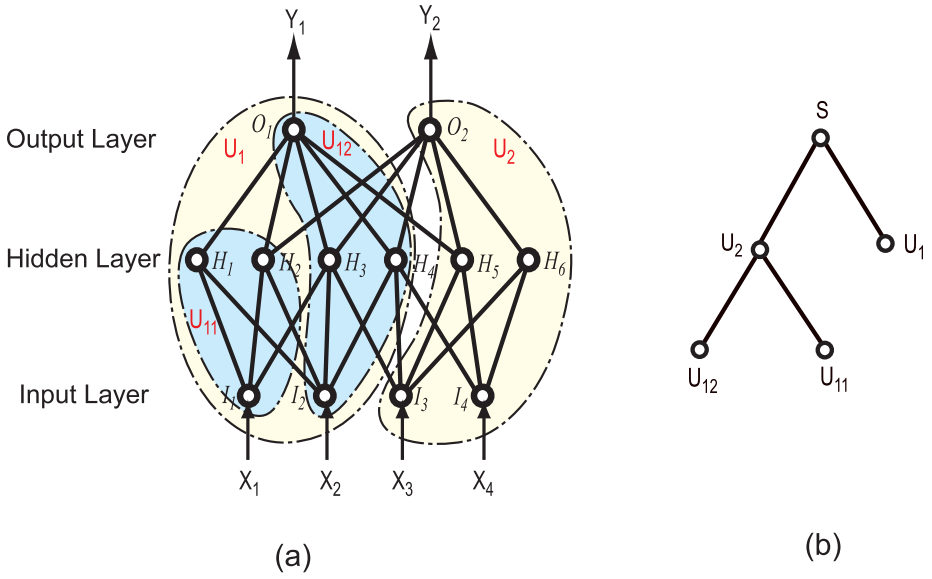


Figure 2.9: Structures of neural networks.

Definition 5 (Feasible Decomposition) *A feasible decomposition is a decomposition which satisfies all possible constraints and in which the sizes of leave modules in the hierarchical tree should be no larger than the threshold, and the sizes of their parents are larger than the threshold.*

2.5.6 Comparative Nature

It is common to say "system A is more modular than B" such and such. This requires that the formally defined modularity should be comparative. Although "modular" is a very fuzzy concept, it is feasible to define measurable modularity, just like "high" and "height."

It has been seen in survey Section 2.4 that some views thought that modularity should be limited to modular structures. Because of the following three reasons, the better way is to extend modularity to every system. Firstly, "modular" and "modular structure" are fuzzy. There is no clear separation between modular structures and non-modular structures. Then, if modular structures are assumed to have modularity and non-modular structures not, modularity would also be a fuzzy concept. The fuzziness and informality are not what we need. Secondly, it is common to say "system A is more modular than B" such and such. This requires formally defined modularity should be comparative and needs a quantitative parameter to characterize modularity. Thirdly, the reason to study modularity is to propose a formal measure of modularity and then to evolve a low modular system to a high modular one. If only modular structures have modularity and non-modular structures have 0 modularity, there are sharp jumps in evolution landscapes which we attempt to avoid since it is more difficult to evolve in such a discontinuous landscape than smooth one. If every system has modularity, then evolution landscapes are smoothed, as shown in Figure 2.10.

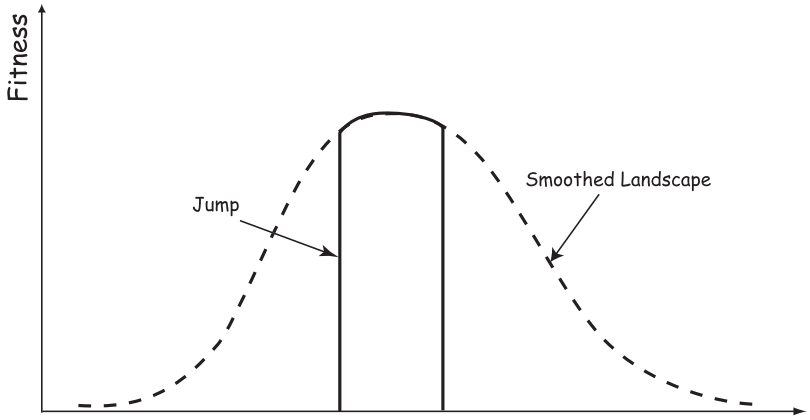


Figure 2.10: Different types of landscapes.

So, modularity should be universal in the sense that modularity is a quantitative parameter of every system. The extreme cases of modularity is a completely integrative system, which has the lowest modularity, and fully decomposable systems, which have the highest modularity.

2.5.7 Definition of Modularity

In summary, modularity has the following characteristics:

1. Hierarchy. The modularity of a system is an aggregation of the modularity at different levels.
2. Globality. Modularity is a global characteristic of a system and is an integration of the modularity of its subsystems.
3. Multi-Dimensionality. Modularity is related to many different aspects such as physical structure, logical structure, and temporal relationships.
4. Relativity: Modularity compares inter-module interaction "density" to intra-module interaction "density."
5. Universality. Modularity is defined over every system.

The modularity of a modular structure (fully decomposable) can be defined as,

Definition 6 (Modularity of a Decomposed System) *Modularity is an attribute describing the degree of overall relative coupling among the parts of the decomposition at different levels in different dimensions.*

From the above discussion, a non-decomposed system plus an imaginary decomposition can be treated a decomposed system, so modularity of a non-decomposed system can be defined on the basis of the definition of modularity of a decomposed system (decomposition). Let us denote modularity

of a decomposition of a system as M_d and C as the set of all feasible decompositions of the system. Then

Definition 7 (Modularity of a System) *Modularity of a system M_m is defined as the maximum of the modularity M_d of all possible decompositions d in C. That is,*

$$M_m = \max_{d \in C} M_d. \tag{2.9}$$

2.6 Module

Modules in a system are the outcomes of increasing modularity of the system, so modules appear in highly modular (decomposable) systems. Since the purpose of defining modules is to make some parts of a large complex system treated nearly independently and therefore easier to analyze or design the whole system, it is necessary to have interfaces to make modules nearly independent from other parts of the system. Sometimes people want to re-use them in other complex systems, which requires that the interfaces should be standardized to be interchangeable in many different environments. Combining these points, a module can be defined as:

Definition 8 (Module) *A module is a unit of a system which is nearly independent of the context and interacts with other units by interfaces.*

2.6.1 System Associated

The concept of module only makes sense under a specific system, i.e., modules should be associated with some system. It is not clear to say "something is a module" without contexts, and it is better to say "something is a module of some system" or "something is a product or system." For example, considering an automobile as a system, a gearbox should be a module of the automobile system, but for manufacturers of the gearbox, they will prefer to think it as a kind of product or system. A module could become a system if it is isolated from the system.

2.6.2 Interface

Interfaces describe in detail how modules interact, how they fit together, and how they communicate. Specification of component interfaces could include: attachment, spatial, transfer, control and communication, environmental, ambient, and user interfaces [84].

Interfaces clearly separate module from other parts of a system and minimize couplings between modules. Therefore, they can be designed, manufactured, and maintained nearly independently. Interfaces need to be designed and specified, and usually this kind of design and specification activities increase the modularity of a system.

2.6.3 Functionality

Modules are usually designed, manufactured, and tested independently. To achieve the independence, it is necessary to provide clear design requirements and well-defined physical behavior (functions). So, a module generally realizes one or a few of the functions of the overall function of the system.

In system synthesis, what designers are generally required to do is to create mechanisms that function as desired or design a system to perform some functions. Pahl [71] points out that it is useful to apply *functions* to describe and solve design problems. Functional modeling provides a direct method for understanding and representing an overall artifact function without reliance on physical structures. Otto and Wood [70] also argued that conceptualizing, defining, or understanding an artifact, product, or system in terms of function is a fundamental aspect of engineering design. At the very beginning phase of design, there is only function structure and no physical realization,[6] so in order to exploit modularity and identify modules in the initial conceptual design or reverse engineering, which reduces many efforts, development time, and costs for product design [71], it is useful to decompose conceptual designs according to function structures and, therefore, later physical designs.

2.6.4 Interchangeable

It is not necessary that modules be interchangeable. Interchangeability requires that interface specifications of modules should be standard. If a component is just developed and is new to the industry, then its interfaces are bound to be ill-designed and specified, and even if the component is unique to a company, then its interface specifications are generally not standardized within the industry. Only when specifications of a component become well specified and standardized do they become a standard component.

2.7 Summary

As pointed out in Chapter 1, modularity or modular structures are commonly existing in complex natural and artificial systems. This implies that modular structures are advantageous to non-modular structures for complex systems. This chapter begins with a discussion on the benefits of modularity, but modularity is not free, and it also has costs.

Based on the surveys on existing views on modularity, several questions about modularity have been clarified. Specifically they are system vs. model, specific meanings of "inter-module" and

[6] According to Pahl and Beitz [71], the design process contains four phases: 1) Product planning and clarifying the task; 2) Conceptual design: establishing function structures; 3) Embodiment design:developing a working structure; 4) Detail design: optimizing the design parameter.

"intra-module," relativity of couplings, aggregation of modularity at different levels in a hierarchical structure, application domain of definition, and how to quantify couplings. Except for coupling quantification, the other 5 questions have been discussed in this chapter. The remaining question on coupling quantification will be discussed in the next two chapters.

For a nearly-decomposable system (or decomposition), modularity describes the degree of overall relative couplings between parts at different scales in different dimensions. For a not-fully-decomposable system, modularity is defined as the maximal modularity of all feasible decompositions of the system.

Finally, the concept of module has been discussed. A module is viewed as a unit of a system which is nearly independent of the contexts and interacts with other units by interfaces.

Chapter 3

Quantifying Couplings with Information-Theoretic Methods

3.1 Introduction

There are many qualitative and exploratory studies on modularity [3, 28, 46, 61, 81, 85, 86, 100], but few are quantitative [48, 29, 63]. Existing modular product design methods can be classified into the following two categories: function-based methods [71, 94] and matrix-based methods [19, 91, 27, 29, 113, 89]. Pahl and Beitz [71] developed a formal function model and proposed that modular product architectures can be derived from function diagrams which describe the flow of energy, materials, and signals between subsystems. Based on formal function models described by Pahl and Beitz, Stone et al.[94] developed a set of three heuristic methods for identifying modules from the function models. Most function-based methods are heuristic, and not quantitative, methods. The design structure matrix (DSM) representation method was invented by Donald Steward [93] and extended and refined by Steven Eppinger [19]. Most matrix-based modular product design methods are based on some quantitative measures of modularity. Most of those quantitative modularity measures typically use linkage counting methods, which compare the average number of linkages between modules with the average number of linkages inside modules. For example, following is one typical modularity measure of a product [34]:

$$\text{Modularity} = \frac{\sum_{k=1}^{M}\left(\frac{\sum_{i=n_k}^{m_k}\sum_{j=n_k}^{m_k}R_{ij}}{(m_k-n_k+1)^2} - \frac{\sum_{i=n_k}^{m_k}(\sum_{j=1}^{n_k-1}R_{ij}+\sum_{j=m_k+1}^{N}R_{ij})}{(m_k-n_k+1)(N-m_k+n_k-1)}\right)}{M},$$

where n_k and m_k are indices of the first and last component in k^{th} module, M is the total number of modules inside the product, N is the total number of components in the product, and R_{ij} is the value of i^{th} row and j^{th} column element in the modularity matrix.

All of these methods assume that interactions between different function units or design parameters have been quantified. How are the interactions quantified? Generally, they are done according to designers' experience and subjective intuitions, which are not formal. Though the early stages of

design processes need some intuition and an art sense, those quantifications could be done formally in redesign processes and later phases of design processes. According to the author's knowledge, there is no such quantitative way to quantify the interactions.

From information-theoretic views, this chapter develops mutual information-based methods which have been used to quantify independence in independent component analysis [49] to provide quantitative ways to measure interactions. Now that information-theoretic methods can quantify interactions, they can directly quantify modularity without mapping designs or products to DSM or function diagrams first. Most information theoretic concepts, such as entropy and mutual information, are based on uncertainty and randomness very commonly existing in engineering practice and products, which can be modeled as stochastic systems since working conditions for engineering products are stochastic, noises in working environments are random, and the inputs of the system are random.

Since mutual information-based methods are based on randomness, systems of random variables are considered firstly, and a framework of modularity measure based on mutual information is established. Under the framework, modularity of dynamic behaviors of stochastic systems and design processes are discussed. According to information-theoretic views, adjacency matrices of weighted graphs can be mapped to covariance matrices of gaussian random variables, and it can be shown that general linkage counting methods are a special case of the information-theoretic views.

3.2 Preliminary on Information Theory

Most material in this section is from "The Theory of Information and Coding" by R. J. McEliece [62] and "Elements of Information Theory" by T. M. Cover and J. A. Thomas [15]. You can refer to these two books for more details.

3.2.1 Discrete Random Variables

The entropy $H(X)$ of a discrete random variable X with probability distribution $p(x)$ is defined by

$$H(X) = -\sum_{x} p(x) \log p(x). \tag{3.1}$$

The joint entropy $H(X, Y)$ of two discrete random variables X and Y with joint probability distribution $p(x, y)$ is

$$H(X, Y) = -\sum_{x,y} p(x, y) \log p(x, y). \tag{3.2}$$

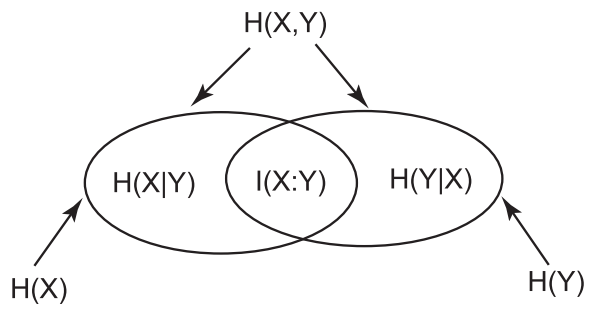


Figure 3.1: Relationship between entropy and mutual information

The conditional entropy $H(Y|X)$ is

$$\begin{aligned} H(Y|X) &= -\sum_{x,y} p(x,y)\log p(y|x) && (3.3)\\ &= H(X,Y) - H(X). && (3.4) \end{aligned}$$

The mutual information $I(X:Y)$ is defined as

$$\begin{aligned} I(X:Y) &= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\ &= H(X) - H(X|Y). \end{aligned}$$

It is not difficult to get:

1. Symmetric: $I(X:Y) = I(Y:X)$.

2. Non-negativity: $I(X:Y) \geq 0$.

   This immediately follows from Jensen's inequality and the fact that $-\log x$ is a convex function:

$$I(X:Y) \geq -\log\sum_{x,y} p(x,y)\frac{p(x)p(y)}{p(x,y)} = -\log 1 = 0.$$

3. $I(X:Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. These relations can be expressed in a Venn diagram shown in Figure 3.1.

3.2.2 Continuous Random Variables

The differential entropy of a set of random variable $X_1, \cdots, X_n$ with density $f(x_1, \cdots, x_n)$ is defined as

$$h(X_1, \cdots, X_n) = -\int f(x_1, \cdots, x_n)\log f(x_1, \cdots, x_n)dx_1 \cdots dx_n. \quad (3.5)$$

Not like discrete case, $h(X)$ is not necessary to be non-negative. For example, let X be a uniform random variable on $[0, a]$. Then

$$h(X) = \int_0^a \frac{1}{a} \log a dx = \log a.$$

When $a < 1$, then $h(X) = \log a < 0$. Unlike $H(X)$ in discrete cases, $h(X)$ is not a quantity to measure the randomness of X [62].

There is no general explicit analytic formula to calculate $h(X)$, but there is one for Gaussian random variables (vectors). Given a set of random variables $S = \{X_1, X_2, \cdots, X_n\}$, which have a multivariate normal distribution with mean μ and covariance matrix K, and the probability density function

$$f(\mathbf{x}) = \frac{1}{(\sqrt{2\pi})^n \sqrt{|K|}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu)^T K^{-1}(\mathbf{x} - \mu)\},$$

where $|K|$ denotes the determinant of K, then the differential entropy of S is

$$h(S) = \frac{1}{2} \log(2\pi e)^n |K|. \tag{3.6}$$

The mutual information $I(X : Y)$ between two random variables with joint density $f(x, y)$ is defined as (refer to [62] for more formal definition):

$$I(X : Y) = \int f(x, y) \log \frac{f(x, y)}{f(x)f(y)} dxdy \tag{3.7}$$

From the above definition,

$$\begin{aligned} I(X : Y) &= \int f(x, y) \log f(x|y) dxdy - \int f(x) \log f(x) dx \\ &= h(X) - h(X|Y) \qquad (3.8) \\ &= h(X) + h(Y) - h(X, Y) \qquad (3.9) \end{aligned}$$

The last step derivation in the above equation is due to $h(X|Y) = h(X, Y) - h(Y)$. We must be careful to use this result if any of the differential entropies is infinite. It is easy to see that $I(X : Y)$ is nonnegative by the following inference:

$$I(X : Y) = -\int f(x, y) \log \frac{f(x)f(y)}{f(x, y)} dxdy \geq \log \int f(x)f(y) dxdy = \log 1 = 0,$$

where the inequality comes from Jensen's inequality.

3.3 System of Random Variables

3.3.1 Quantifying Interactions

Firstly, when we talk about coupling between two random variables, the first quantity coming into consideration could be the covariance of the two random variables. Unfortunately, the covariance is not the whole. Even if two random variables have zero covariance, they could still have higher-order correlations when they are non-gaussian and, therefore, non-zero coupling. So, couplings should be quantified by some measure which includes high-order correlations. According to information-theoretic views, coupling means how much uncertainty (freedom) of the state of one random variable can be gotten away with if the state of the other one is known. This is actually the mutual information between the two random variables.

Let us consider quantifying couplings under the case which includes observers (designers in design processes), as shown in Figure 3.2. Suppose there is a set of random variables $S = \{X_1, X_2 \cdots, X_n\}$, which is separated into two subsystems with subsets S_1 and S_2, respectively, and there are two observers O_1 and O_2 measuring the two subsystem, respectively. From an observer's view, the interaction of S_1 on S_2 is how much information the observer can infer about S_2 if he has observed information O_1 of S_1. In engineering design, S_is are usually fully observable since observers (designers) know designs in design evolution, as discussed in chapter 1. That is, each observer can completely decide the state (information) of the corresponding subsystem, i.e., $H(S_i|O_i) = 0, \quad i = 1, 2$. Then, $I(O_1 : S_2)$ gives the information the observer 1 (O_1) can infer about the subsystem S_2 based on its observations on the subsystem S_1, and similarly, $I(O_2 : S_1)$ means how much O_2 knows about S_1 while it only observes S_2. When the two subsystems totally decoupled, O_1 can not infer any information about S_2 and neither can O_2 infer any information about S_1. That is, $I(O_1 : S_2) + I(O_2 : S_1) = 0$. On the other extreme case, where S_1 and S_2 are totally coupled, then O_1 and O_2 can completely determine the states of S_2 and S_1, respectively, and $I(O_1 : S_2)$ and $I(S_1 : O_2)$ are maximized. In the intermediate state between completely uncoupled and totally coupled, the stronger interaction of S_1 on S_2, the larger $I(O_1 : S_2)$ and $I(O_2 : S_1)$. This suggests the following definition of interaction of one set of random variables on the other random variables, denoted as C_{ij}:

$$C_{ij} = \min_{(O_i) \in V} I(O_i : S_j), i, j = 1, 2,$$

where $V = \{(O_i) : H(S_j|O_i) = 0\}$. This definition gives some difficulty to compute since the set V is not totally determined by the system S, and it is not very clear what elements V have. It can be simplified and become computable by the following result.

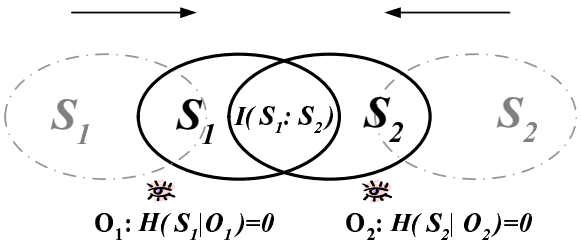


Figure 3.2: Mutual information as a measure of coupling of two systems of random variables.

Theorem 1 *The interaction of S_i on S_j is*

$$C_{ij} = I(S_i : S_j). \tag{3.10}$$

Proof:

$$I(S_i : S_j|O_i) = H(S_i|O_i) - H(S_i|S_j, O_i) = 0, \text{ since } H(S_i|O_i) = 0$$

We can write $I(S_i : S_j|O_i)$ in an other way:

$$I(S_i : S_j|O_i) = H(S_j|O_i) - H(S_j|S_i, O_i) = 0.$$

Therefore,

$$H(S_j|S_i) - H(S_j|O_i) = H(S_j|S_i) - H(S_j|S_i, O_i) = I(S_j; O_i|S_i) \geq 0.$$

The last inequality holds because the negativity of mutual information. Then we have

$$H(S_j|O_i) \leq H(S_j|S_i).$$

Therefore,

$$I(O_i : S_j) \quad = \quad H(S_j) - H(S_j|O_i) \geq H(S_j) - H(S_j|S_i) = 2I(S_1 : S_2).$$

That is, $I(O_1 : S_2) + I(O_2 : S_1)$ has a lower bound $2I(S_1 : S_2)$, and obviously this bound can be achieved by setting $O_1 = S_1$ and $O_2 = S_2$. So, $C = I(S_1 : S_2)$; ■

Since $I(S_i : S_j)$ is symmetric, i.e., $I(S_i : S_j) = I(S_j : S_i)$, interactions between two sets of random variables are symmetric and non-directional.

The following example will consider multivariate or bivariate normal random variables. Normal

distributions are common in engineering, mainly due to the Central Limit Theorem, which states that given a set of N-independent random variables $\{X_1, X_2, \cdots, X_N\}$, the random variable $X = \sum_{i=1}^{N} X_i/N$ approaches a normal distribution as N increases.

By the following result from [65], a subset of a set of multivariate normal random variables is still normal. Let $\mathbf{X}$ be an n-dimensional normal random vector with mean μ and covariance matrix K. Let A be an $m \times n$ matrix of rank m. Then, $\mathbf{Y} = A\mathbf{X}$ has an m-dimensional normal random vector with mean β and covariance matrix Q given, respectively, by

$$\beta = A\mu \quad \text{and} \quad Q = AKA^T. \tag{3.11}$$

To see that a subset is still normal, consider the following example: to extract the subset $(X_1, X_3, X_4)^T$, the following A can be used to get the desired elements directly:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}.$$

Now suppose S is separated into two subsets S_1 and S_2, and we come to calculate the mutual information $I(S_1 : S_2)$. From the above result, S_1 and S_2 both have multivariate normal marginal distributions. Without the loss generality, it can be assumed that $S_1 = \{X_1, X_2, \cdots, X_i\}$ and $S_2 = \{X_{i+1}, X_{i+2}, \cdots, X_n\}$. K is symmetric positive definite and can be represented as

$$K = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix},$$

where A is an $i \times i$ matrix, B is an $i \times (n-i)$ matrix, and C is an $(n-i) \times (n-i)$ matrix.

Then, by Equation 3.11, S_1 has mean $\mu^{\mathbf{1}} = \{\mu_1, \cdots, \mu_i\}$ and covariance matrix A, and S_2 has mean $\mu^{\mathbf{2}} = \{\mu_{i+1}, \cdots, \mu_n\}$ and covariance matrix C. Obviously, A and C are symmetric positive definite. Therefore, by Equation 3.6,

$$h(S_1) = \frac{1}{2}\log(2\pi e)^i |A|, \text{ and } h(S_2) = \frac{1}{2}\log(2\pi e)^{n-i}|C|.$$

Then, by Equation 3.9,

$$\begin{aligned} I(S_1 : S_2) &= h(S_1) + h(S_2) - h(S_1, S_2) \\ &= \frac{1}{2}\log(2\pi e)^i |A| + \frac{1}{2}\log(2\pi e)^{n-i}|C| - \frac{1}{2}\log(2\pi e)^n |K| \\ &= \frac{1}{2}\log\frac{|A| \cdot |C|}{|K|}. \end{aligned} \tag{3.12}$$

Let $D = -A^{-1}B$, and compute

$$\begin{aligned} |K| &= \left|\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}\right| = \left|\begin{bmatrix} I & 0 \\ D^T & I \end{bmatrix}\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}\begin{bmatrix} I & D \\ 0 & I \end{bmatrix}\right| \\ &= \left|\begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix}\right| = |A| \cdot |C - B^T A^{-1} B|. \end{aligned}$$

Since the determinant of matrix $M = [m_{ij}]$ is continuous with respect to the elements, m_{ij}, of matrix M, as $B \to 0$ in the sense that every elements in B approaches 0, then $|C - B^T A^{-1} B|$ will approach $|C|$ and

$$\frac{|A| \cdot |C|}{|K|} = \frac{|C|}{|C - B^T A^{-1} B|} \to 1,$$

and, therefore, by Equation 3.12,

$$I(S_1 : S_2) \to 0 \quad \text{as} \quad B \to 0.$$

This shows that generally $I(S_1 : S_2)$ is "continuous" with respect to the strength of coupling in some sense.

Note that matrix B represents the covariance between elements in S_1 and those in S_2. When the coupling between S_1 and S_2 is weak, then the elements in B are small, and therefore $I(S_1 : S_2)$ is small.

Example: Let us consider bivariate normal random variables X_1, X_2, X_3, and X_4 with covariance matrix

$$K = \sigma \begin{bmatrix} 1 & \rho & \epsilon & \epsilon \\ \rho & 1 & \epsilon & \epsilon \\ \epsilon & \epsilon & 1 & \rho \\ \epsilon & \epsilon & \rho & 1 \end{bmatrix}.$$

ρ can be considered the intra-module interaction, and ϵ represents the inter-module interaction if X_1, X_2, X_3, and X_4 are clustered into two module $S_1 = (X_1, X_2)$ and $S_2 = (X_3, X_4)$, as shown in Figure 3.3. By Sylvester criterion [44] that a matrix is positive definite if and only if its leading principal minors have positive determinants, K must satisfy the following conditions to be a covariance

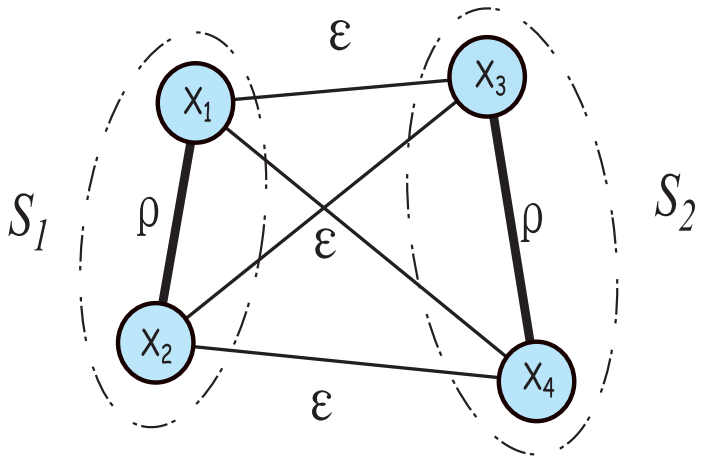


Figure 3.3: Random system of example 3.3.1.

matrix:

$$1-\rho^2 > 0 \tag{3.13}$$

$$1-\rho^2-2(1-\rho)\epsilon^2 > 0 \tag{3.14}$$

$$1-2\rho^2+\rho^4-4(1-\rho)^2\epsilon^2 > 0 \tag{3.15}$$

This gives

$$-1<\rho<1, \quad -\frac{\rho+1}{2}<\epsilon<\frac{\rho+1}{2}.$$

Now let us consider the following two-way clustering: $S_1=\{X_1,X_2\}$ and $S_2=\{X_3,X_4\}$. Since σ will not affect the mutual information, let $\sigma=1$. Then,

$$I(X_1:X_2)=I(X_3:X_4)=-\log\sqrt{(1-\rho^2)},$$

which is plotted in Figure 3.4. It is easy to note that there is no coupling, i.e. $I(X_1:X_2)=0$ when $\rho=0$, i.e., X_1 and X_2 are independent, and the figure is symmetric with respect to ρ. By Equation 3.12:

$$I(S_1:S_2)=\frac{1}{2}\log\frac{(1-\rho^2)^2}{1-2\rho^2+\rho^4-4(1-\rho)^2\epsilon^2}=-\frac{1}{2}\log\left(1-4\left(\frac{\epsilon}{1+\rho}\right)^2\right).$$

The relation between $I(S_1:S_2)$ and ρ, ϵ is shown in Figure 3.5. ■

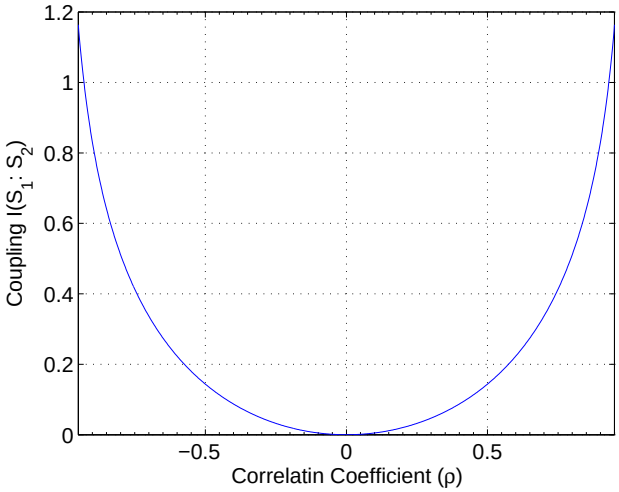


Figure 3.4: The coupling between bivariate normal variables.

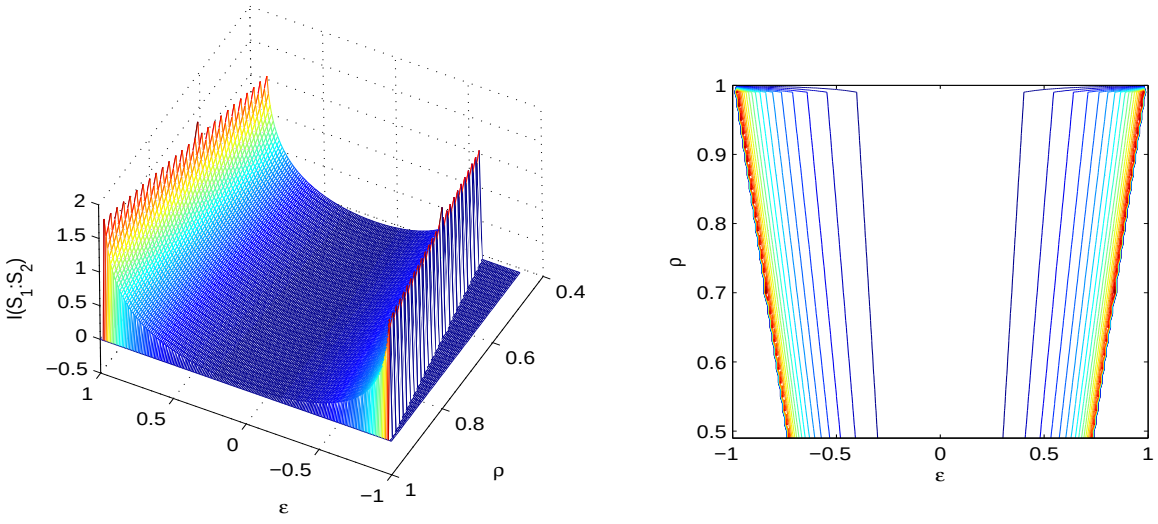


Figure 3.5: $I(S_1 : S_2)$ vs. ρ and ϵ: the right is contour plot.

3.3.2 Modularity Measure

3.3.2.1 Two Clusters and One Level

Once coupling can be quantified, modularity can also be measured quantitatively. Firstly, let us consider the simplest case that the system $S = \{X_1, X_2, \cdots, X_n\}$ is separated into two cluster $S_1 = \{X_1, \cdots, X_i\}$ and $S_2 = \{X_{i+1}, \cdots, X_n\}$. From the discussion in Chapter 2 briefly , modularity is to measure the relative strength of inter-module interactions (coupling) compared to intra-module interactions (cohesion). Inter-module coupling can be quantified by mutual information $I(S_1 : S_2)$, but how are intra-module interactions quantified? Basically, intra-module interactions are the couplings between the elements inside a module. If you consider S_1 or S_2 as a subsystem, then the intra-module interaction should be the coupling among the elements. Here, assume the clustering has only one level. That is, there is no clustering inside S_1 and S_2. Then, the coupling among the elements in S_1 can be measured as

$$C_1^{in} = \sum_{k=1}^{i} \sum_{j=k+1}^{i} I(X_k : X_j).$$

There are totally $i(i-1)/2$ terms in above formula. Due to the relativity of interaction discussed in Section 2.5.3, the coupling needs to be averaged out. Then,

$$\bar{C}_1^{in} = \frac{2}{i(i-1)} \sum_{k=1}^{i} \sum_{j=k+1}^{n} I(X_k : X_j).$$

Now, $I(S_1 : S_2)$ can be normalized by the number N^{out} of possible interactions between S_1 and S_2. Since S_1 and S_2 have i and $n-i$ elements respectively, $N_{12}^{out} = i(n-i)$. The average intermodule coupling is

$$\bar{C}_{12}^{out} = \bar{I}(S_1 : S_2) = \frac{I(S_1 : S_2)}{N_{12}^{out}} = \frac{I(S_1 : S_2)}{i(n-i)}.$$

The overall coupling of the clustering is defined as linear average

$$C^1 = \frac{1}{2}\left(\frac{\bar{C}_{12}^{out}}{\bar{C}_1^{in}} + \frac{\bar{C}_{21}^{out}}{\bar{C}_2^{in}}\right).$$

When non-linear aggregation in [68, 69, 87] is used,

$$C^1 = \left(\frac{1}{2}\left(\frac{\bar{C}_{12}^{out}}{\bar{C}_1^{in}}\right)^s + \frac{1}{2}\left(\frac{\bar{C}_{21}^{out}}{\bar{C}_2^{in}}\right)^s\right)^{\frac{1}{s}}, \qquad s \in [-\infty, \infty],$$

where $s = -\infty, 0$, and ∞ cases are considered as limits, and they correspond to min, geometric average, and max, respectively. Then, modularity M can be defined as the inverse of the overall coupling since the stronger the overall coupling, the lower the modularity.

The basic idea of above computation is the following. The intuitive meaning of mutual information $I(S_1 : S_2)$ is to measure how much information on S_1 can be inferred from the known information on S_2. If it needs to be normalized to a relative quantitative variable, it is natural to normalize it by the amount of information S_1 has. Another way to calculate the information inside S_1 is to use generalized mutual information [49, 97]. The mutual information of a set of random variables $S = \{X_1, X_2, \cdots, X_n\}$ is the Kullback-Leibler distance between $p(x_1, x_2, \cdots, x_n)$ and $\prod_{k=1}^{n} p(x_k)$, i.e.,

$$\begin{aligned} I(S) = I(X_1, \cdots, X_n) &= \int p(x_1, \cdots, x_n) \log \frac{p(x_1) \cdots p(x_n)}{p(x_1, \cdots, x_n)} dx_1 \cdots dx_n \\ &= \sum_{k=1}^{n} H(X_k) - H(X_1, \cdots, X_n), \end{aligned}$$

which is non-negative according to the Jensen inequality. Then, the averaged mutual information is

$$\bar{I}(S_1) = \frac{2I(S_1)}{i(i-1)}, \quad \bar{I}(S_2) = \frac{2I(S_2)}{(n-i)(n-i-1)} \tag{3.16}$$

The overall coupling of a two-cluster decomposition is defined as

$$C^2 = \frac{1}{2}\left(\frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)} + \frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_2)}\right), \tag{3.17}$$

or using non-linear aggregation,

$$C^2 = \left(\frac{1}{2}\left(\frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)}\right)^s + \frac{1}{2}\left(\frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_2)}\right)^s\right)^{\frac{1}{s}}, \qquad s \in [-\infty, \infty]. \tag{3.18}$$

Then, modularity is inverse to coupling

$$M^1 = \frac{1}{C^1} \quad \text{and} \quad M^2 = \frac{1}{C^2}.$$

Example: Let's continue example 3.3.1. Assume the linear aggregation is used in this example. Then

$$C^1 = C^2 = \frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)} = \frac{4\log\left(1 - 4\left(\frac{\epsilon}{1+\rho}\right)^2\right)}{\log(1-\rho^2)}.$$

The relation between C^1 and ϵ, ρ is shown in Figure 3.6. Now let's consider some asymptotic behaviors of $M^1 = 1/C^1$. For any fixed $\rho \in (-1, 1)$, as $\epsilon \longrightarrow \pm\frac{1+\rho}{2}$, $M^1 \longrightarrow 0$. For any fixed

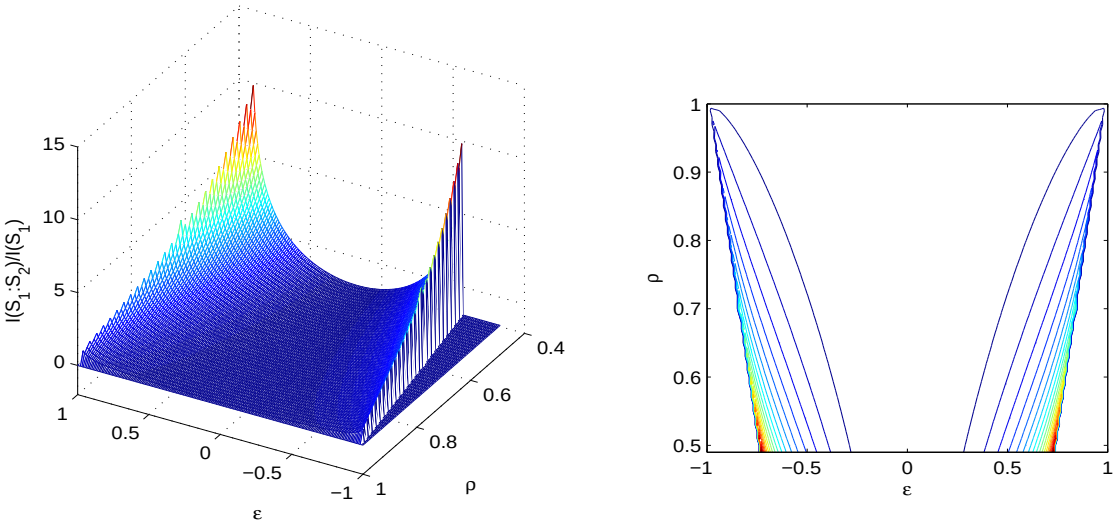


Figure 3.6: C^1 vs. ϵ and ρ.

$\epsilon \in (-0.5, 0.5)$, as the intra-module interaction approaches zero, $\rho \longrightarrow 0$, then the modularity should be very low, $M^1 \longrightarrow 0$, no matter what positive value the inter-module interaction has. In Figure 3.6, the larger the ρ, the weaker effects ϵ has on M^1. This is consistent with our intuition that modularity is affected the relative inter-module interactions.

■

3.3.2.2 Multi-cluster and One Level

It is easy to generalize the idea for the case having two clusters and one level. Suppose there are m subsystems (modules), and each subsystem S_i has r_i elements, $i = 1, \cdots, m,$. Then,

$$C^1 = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \frac{\bar{C}_{ij}^{out}}{\bar{C}_i^{in}} \tag{3.19}$$

and

$$C^2 = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \frac{\bar{I}(S_i : S_j)}{\bar{I}(S_i)}. \tag{3.20}$$

When non-linear aggregation is used,

$$C^1 = \left(\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \left(\frac{\bar{C}_{ij}^{out}}{\bar{C}_i^{in}} \right)^s \right)^{\frac{1}{s}}, \qquad s \in [-\infty, \infty] \tag{3.21}$$

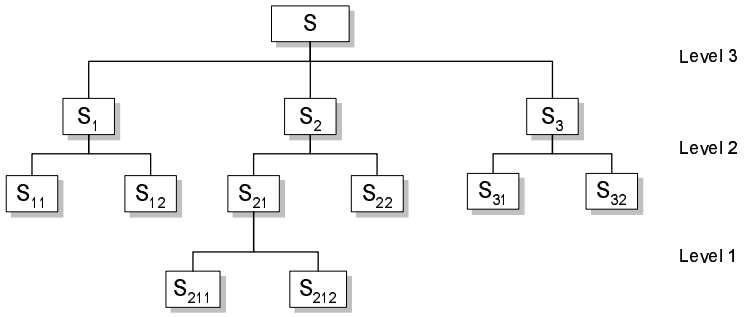


Figure 3.7: A hierarchical structure.

and

$$C^2 = \left(\frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \left(\frac{\bar{I}(S_i : S_j)}{\bar{I}(S_i)} \right)^s \right)^{\frac{1}{s}}, \qquad s \in [-\infty, \infty]. \tag{3.22}$$

Then modularity is inverse to coupling

$$M^1 = \frac{1}{C^1} \quad \text{and} \quad M^2 = \frac{1}{C^2}. \tag{3.23}$$

3.3.2.3 Multi-cluster and Multi-level

In multi-level hierarchy, the average modularity is first calculated at every level. For example, consider to calculate the modularity at level 2 in Figure 3.7. There are three random subsystems: $S_1 = \{\{S_{11}\}\{S_{12}\}\}$, $S_2 = \{\{S_{21}\}\{S_{22}\}\}$, and $S_3 = \{\{S_{31}\}\{S_{32}\}\}$. The modularity of each subsystem is calculated without considering the modules inside the subsystems. That is, assume that every subsystem is flat without hierarchical structures. For example, when computing modularity of S_2 at level 2, the decomposition inside S_{21} will not be considered. Then, the modularity of subsystems at the same level is averaged to get the modularity M_i of the level i. Now aggregation methods introduced in Chapter 2 can be used to aggregate the modularity at different levels to get the modularity of the system (decomposition).

3.3.2.4 Modularity of a System

Let $\mathcal{C}$ be the set of all feasible clustering ways. Then the modularity of the system, M, is calculated as

$$M = \max_{c \in \mathcal{C}} M_c$$

3.3.3 Mutual Information in Non-gaussian Cases

The calculation of mutual information of multi-gaussian random vectors is given in Section 3.3. How do we then calculate mutual information for non-gaussian random vector $\mathbf{Y}$? Unfortunately, there is no explicit analytic formula in this case. So, we want to approximate joint probability density $p(\mathbf{y})$ with a m dimensional gaussian density $\phi(\mathbf{y})$ with the same covariance matrix as $p(\mathbf{y})$. Suppose

$$p(\mathbf{y}) = \phi(\mathbf{y})(1 + \delta(\mathbf{y})). \tag{3.24}$$

Then

$$\begin{aligned} H(p) &= -\int p \log p d\mathbf{y} \\ &= -\int p \log \phi d\mathbf{y} - \int p \log \frac{p}{\phi} d\mathbf{y} \\ &=^{(1)} -\int \phi \log \phi d\mathbf{y} - \int p \log \frac{p}{\phi} d\mathbf{y} \\ &= H(\phi) - \int \phi(1+\delta)\log(1+\delta) d\mathbf{y} \\ &\approx^{(2)} H(\phi) - \int \phi(1+\delta)(\delta - \frac{\delta^2}{2}) d\mathbf{y} \\ &\approx H(\phi) - \int \phi(\delta + \frac{\delta^2}{2}) d\mathbf{y}, \end{aligned} \tag{3.25}$$

where (1) follows from the fact that $\log \phi$ only contains the first and second order of $\mathbf{y}$, and p,ϕ have the same means and covariance matrices; (2) comes from the approximation $\log(1+y) \approx y - \frac{y^2}{2}$.

Now we need a good approximation for $\delta(\mathbf{y})$. The multivariate Edgeworth expansion of $p(\mathbf{y})$ gives an approximation of $\delta(\mathbf{y})$. One approximation up to order five is given by [6]

$$\begin{aligned} \delta(\mathbf{y}) &= \frac{1}{3!}\kappa_{ijk}H_{ijk}(\mathbf{y}) + \frac{1}{4!}\kappa_{ijkl}H_{ijkl}(\mathbf{y}) \\ &\quad + \frac{10}{6!}\kappa_{ijk}\kappa_{lpq}H_{ijklpq}(\mathbf{y}) + O(n^{-3/2}) \\ &\approx \frac{1}{3!}\kappa_{ijk}H_{ijk}(\mathbf{y}), \end{aligned} \tag{3.26}$$

where κ_{ijk} is the third cumulant of $\mathbf{Y}$ and $H_{ijk}(\mathbf{y})$ is the ijk-th Hermite polynomial. Generally, $H_{i_1 i_2 \cdots i_v}(\mathbf{y})$, where $\mathbf{y} = (y_1, \cdots, y_m)$, is defined on the basis of $H_k(y)$. Let $\{j_1, j_2, \cdots, j_u\}$ be the different elements of $\{i_1, i_2, \cdots, i_v\}$, and each j_k has n_k repetitions, $1 \le k \le u$. Then, $H_{i_1 i_2 \cdots i_v}(\mathbf{y}) = H_{n_1}(y_{j_1})H_{n_2}(y_{j_2}) \cdots H_{n_u}(y_{j_u})$. For example, $H_{iik}(\mathbf{y}) = H_2(y_i)H_1(y_k)$.

Plug Equation 3.26 into Equation 3.25, and note that $\int \phi\delta d\mathbf{y} = 0$ and the orthogonality properties

of the Hermite polynomials. We have

$$H(p) = H(\phi) - \frac{1}{12}\left(\sum_{i=1}^{m}\kappa_{iii}^2 + 3\sum_{i,j=1,i\neq j}^{m}\kappa_{iij}^2 + \frac{1}{6}\sum_{i,j=1,i<j<k}^{m}\kappa_{ijk}^2\right).$$

For the univariate case, entropy can be approximated as [49]

$$p(y) \approx \phi(y)\left(1 + \frac{1}{3!}\kappa_3(y)H_3(y) + \frac{1}{4!}\kappa_4(y)H_4(y)\right) \tag{3.27}$$

$$H(p) \approx H(\phi) - \left(\frac{1}{12}\kappa_3^2(y) + \frac{1}{48}\kappa_4^2(y) + \frac{7}{48}\kappa_4^4(y) - \frac{1}{8}\kappa_3^2(y)\kappa_4(y)\right) \tag{3.28}$$

3.4 Dynamic Behaviors in Stochastic Systems

In the next three sections, engineering systems will be modeled as systems of random variables, and their interactions can be quantified by mutual information. Therefore, their modularity can be measured in the framework established in the previous section. This section discusses the modularity of dynamic behaviors in stochastic systems.

Usually, engineering products can be modeled as stochastic systems since they have stochastic behaviors, stochastic input (including random noises) from random working environments. From dynamic views, design products can be viewed as mappings, which map inputs from environments and initial states to outputs. For any subsystem of a product, the mapping can also be viewed as one which maps inputs from environments and other subsystems and its initial states to outputs to environments and other subsystems. Let us begin with two simple dynamic subsystems, shown in Figure 3.8. Vector $U_i, i = 1, 2$ represent the inputs, $V_i, i = 1, 2$ represent the outputs of subsystems to the outside environment, and $S_i, i = 1, 2$, represent the state status of subsystem i. W_{ij} represents the outputs from subsystem i to subsystem j.

The simple way to quantify the couplings between dynamic behaviors of two subsystems is degree of freedom (DOF). Given the states of one subsystem, the number of independent states of the other subsystem (DOF) can briefly tell the strength of couplings between the two subsystems. The shortcomings of DOF are 1) they only have integer values, that is, they are $0, 1$ functions and therefore can not quantify the situations between totally independent and fully dependent; 2)usually they are limited to rigid body systems. More general measures can be defined on the basis of real physical interaction strength such as power and forces between different modules. However, it is not a good way by the following four arguments. First, the same value probably has a different significance for different physical properties. Second, physical properties have units and different properties have different units. For example, force can use Newton, while distance uses meter. Even one physical property can use different scale units, such as meter and millimeter for distance. It is difficult to

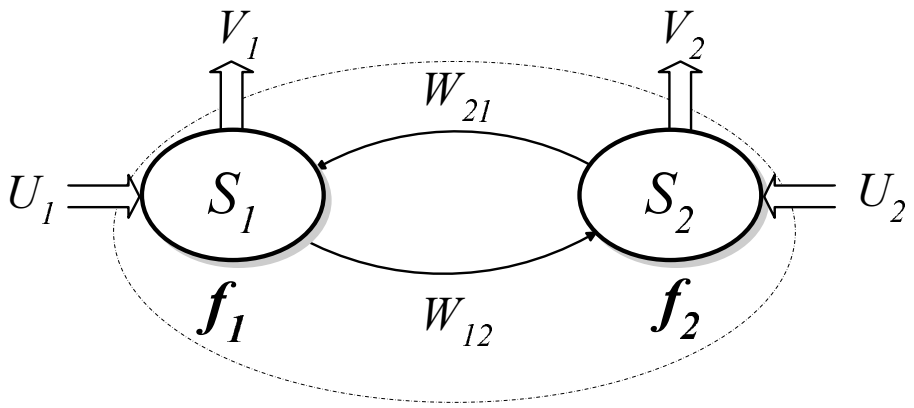


Figure 3.8: A model of two-way clustered stochastic systems

compare different physical values with different units. Third, in some systems, especially dynamic system, very small real values of a physical interaction can dramatically affect the behaviors of the systems [109]. That is, the real value of a physical interaction is not necessary a good indicator of interaction between two subsystems from the view of modularity. The last point is that sometimes the physical types of interactions are not very clear. For example, it is difficult to tell what the physical interaction is between the two particles in the example shown in Figure 3.8 in this section.

The dynamical states of one system can be viewed as random variables. Based on the discussion in the previous section, interactions can be viewed as information flow, which can be quantified by mutual information. Mutual information has no physical units, is applicable to any system which has uncertainty (randomness) and has nice physical semantic meaning. From the view of information flow, if there are strong couplings between two systems, one system's states can be very much inferred from the other system's states, and therefore there is large mutual information between them.

Systems of random variables are relatively simple since the interactions between different units are symmetric and non-directional, and there are no inputs from the outside environment or outputs to the environments, i.e., they are closed systems. The physical stochastic systems are more complex and could be open, that is, there are possible inputs from outside environments and outputs to the environments. Observers in stochastic systems care how the states of one subsystem are affected by the states of the other subsystem and the inputs to the other subsystem from the outside environments. Considering the interactions between systems and environments, the interaction of

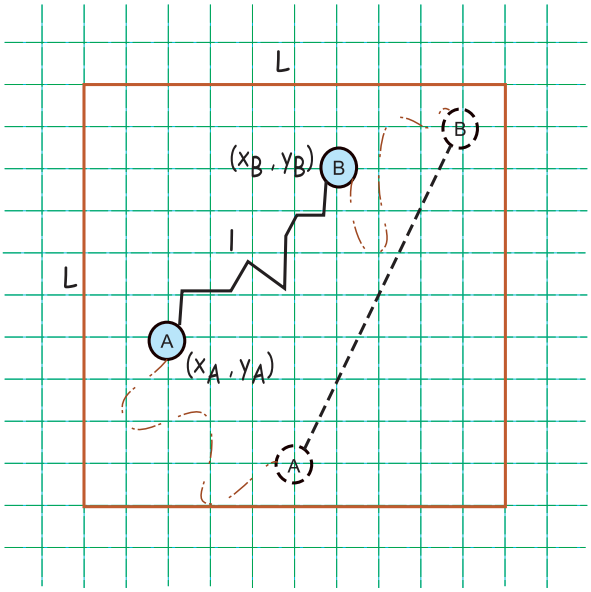


Figure 3.9: Two particles on a lattice.

subsystem $i = 1, 2$ on subsystem $j = 1, 2$ can be defined as

$$C_{ij} = I(S_i : S_j, U_j) = H(S_i) - H(S_i|(S_j, U_j)), \quad i, j = 1, 2, i \neq j. \tag{3.29}$$

According to this definition, interactions in physical stochastic systems are asymmetric and directional. That is, C_{ij} is not necessarily equal to C_{ji}.

By replacing $I(S_1 : S_2)$ and $I(S_2 : S_1)$ by $I(S_1 : S_2, U_2)$ and $I(S_2 : S_1, U_1)$, respectively, in the framework to calculate the modularity a system of random variables, the same framework can also be used to calculate the modularity of dynamic behaviors of stochastic systems.

Let us consider the following simple example, shown in Figure 3.9. Suppose there are two particles A and B randomly moving around in a two-dimensional lattice, and their locations are limited inside the square: $0 \leq x_i \leq 100, 0 \leq y_i \leq 100$. There is a soft link of length l connecting the two particles. Suppose only locations of the two particles, denoted as (X_A, Y_B) and (X_B, Y_B), are concerned. The question is how strong the interaction between the two particles is. One extreme case is when l is larger than the diagonal length of the square. In this case, there is no interaction since the position of particle A does not affect that of particle B and vice versa. The other extreme case is when $l = 0$. In this case, the position of particle B can immediately give that of particle A and vice versa. That is, the two particles have the strongest interaction. How then strong is the interaction under the situation between the two extreme cases?

Assume the prior probability distribution of (X_A, Y_A, X_B, Y_B) is uniform over $(1, \cdots, N)^4$. The posterior probability distribution of (X_A, Y_A, X_B, Y_B) can then be calculated according to the fol-

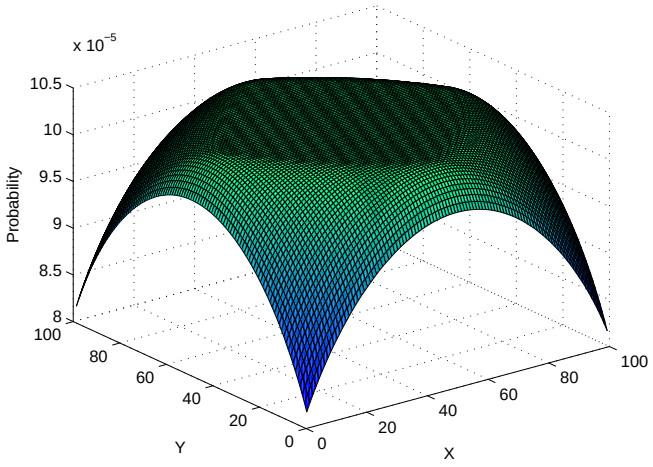


Figure 3.10: Posterior probability of (X_A, Y_A).

lowing constraint:

$$P(x_A, y_A, x_B, y_B) = 0 \quad \text{if } \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} > l.$$

The result of the case where $l = 100$ is shown in Figure 3.10. Then,

$$H(S_1) = \sum_{i=1}^{N} \sum_{j=1}^{N} P(X_A = i, Y_A = j) \log \frac{1}{P(X_A = i, Y_A = j)} = 13.287,$$

and in this simple example, there are no inputs from outside the system, i.e., U_i doesn't appear in the calculation of the coupling C_{ij}. We have

$$H(S_1|S_2) = \sum_{i=1}^{N} \sum_{j=1}^{N} P(X_B = i, Y_B = j) H(S_1|X_B = i, Y_B = j) = 13.252.$$

By symmetry, the couplings

$$C_{21} = C_{12} = H(S_1) - H(S_1|S_2) = 0.0343$$

With the length of link varying from 0 to 500, the change of coupling is shown in Figure 3.11. It is shown in the figure that the coupling decreases exponentially as the length of the link increase, which is consist with our intuition.

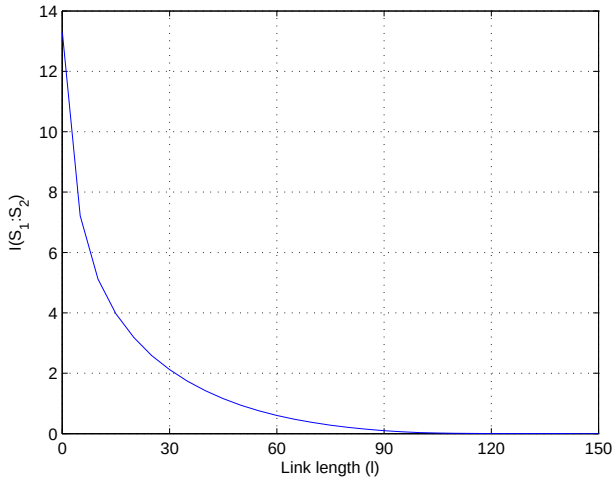


Figure 3.11: Mutual information vs. the length of link connecting the two particles.

3.5 Design Processes and Others

Design process is a search process in a design space, which consists of design parameters. To decouple design tasks means to separate design parameters into different clusters. Design parameters can be viewed as random variables taking values from their domains according to some probability distribution. If there are no specific constraints on the distributions, they can be assumed to be uniform.

Consider the two design tasks, T_A and T_B, in Figure 3.12. They manage design parameters$(X_1, \cdots, X_n)$ and $(Y_1, \cdots, Y_m)$, respectively. The coupling between two design tasks can be viewed as how much information needed to be exchanged to coordinate the two design tasks during the design process. If there is no coupling between two design parameters, P_A and P_B, in T_A and T_B, no information needs to be communicated since the parameters independently take values from their domains. However, if they are totally coupled, it is necessary to communicate the information of P_As value to T_B and the information of P_Bs value to T_A. For the intermediate situations, one task doesn't necessarily send all information he has, but only need to send the amount of information, which is equal to the amount of uncertainty of the other task removed by his decisions. That is, the information needed to be communicated was the mutual information between the two sets of design parameters. Therefore, the coupling between two design parameters can be quantified by the mutual information between them. The framework established in Section 3.3 can also be applied in this case. Here is one simple example.

Example(Fit Interdependency): One simple but general design parameter-interdependency is "fit interdependency" [3], which can arise in the very realm of design that involves tangible artifacts, such as mugs. Let us consider designing a mug with a cap. The design parameters of the vessel could

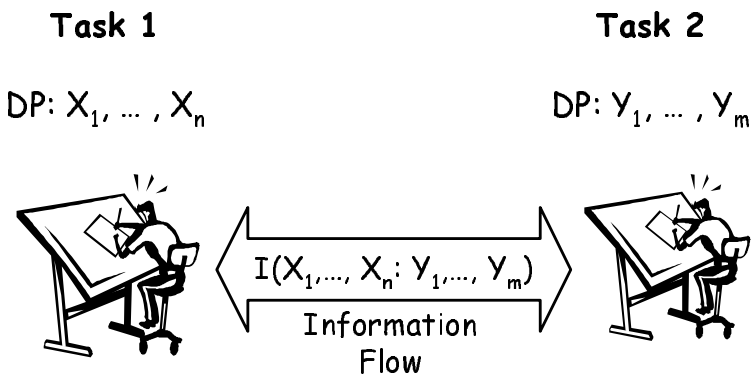


Figure 3.12: How much information need to be communicated?

be diameter(d_v), height(h_v), color(c_v), and material(m_v), and the design parameters of the cap are same and denoted as d_c, h_c, c_c, and m_c, respectively. Obviously, design parameters h_v, c_v, and m_v of the vessel are independent from the design parameters of the cap, and similarly design parameters h_c, c_c, and m_c of the cap are independent from the design parameters of the vessel. [1] Then, the mutual information between two design tasks come down to the mutual information between d_v and d_c. The domains of d_v and d_c are $[5cm, 11cm]$, which are determined by the ergonomics of mugs: if the mug is less than 5 cm or more than 11cm wide, it becomes hard to drink from the vessel. There are no specific constraints on probability distributions, so let's assume d_v and d_c uniformly take values from their domains. Assume the tolerance of design be about 1 mm. Then, the joint probability density of d_v and d_c is shown in Figure 3.13. Let L be the interval length of d_vs (or d_cs) domain and δ be the design tolerance. According to the calculation in Section 3.2, the mutual information between d_v and d_c, $I(d_v : d_c)$, is $\log(L/\delta) = \log(60/1) = 5.9$ bits. When the design tolerance decreases, the mutual information increases, and therefore the coupling between the two design tasks increases. ■

If it is difficult to get probability distributions, coupling can be approximated by the average number of design parameters in one subsystem which could be affected by one design parameter in the other subsystem.

If modularity is considered from manufacturing or life-cycle, it is only necessary to define a set of parameters related to manufacturing or life-cycle and then follow the framework used to quantify couplings in design processes.

[1] It is possible that there are some couplings in those parameters, such as, from the aesthetic view, the colors of vessels and caps should be consistent though they are weak.

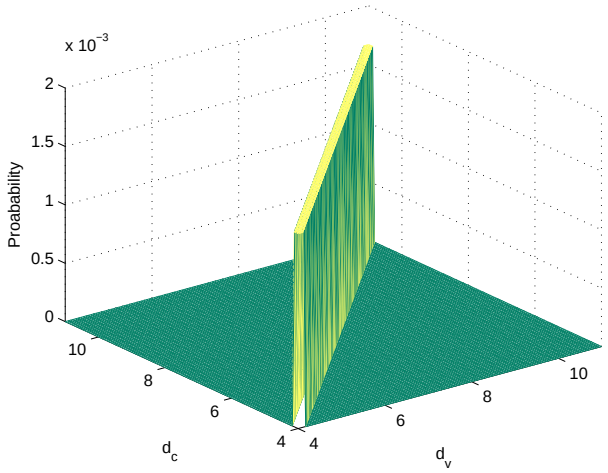


Figure 3.13: Probability distribution of d_v and d_c in example: fit interdependency.

3.6 Information Theoretic Views on DSM and Graphs

Design structure matrices (DSM) [93, 19] can be mapped to a graph where components or design parameters are represented as nodes and the interaction is mapped to an edge with the value of interaction strength as weight. This type of graph is included in a more general graph type, a graph with attributes associated to links. It is feasible to separate the attributes into two categories: unordered attributes and ordered attributes. There is no order between different values of unordered attributes. Generally, unordered attributes are multi-state and used to represent class indices, such as domain attribute in mechanical systems. Ordered attributes are usually used to represent physical quantities. They could take integers or real values. According to the orderness of attributes of links, graphs can be classified into four types: graphs without attributes, graphs only with ordered attributes, graphs only with unordered attributes, and graphs with both unordered and ordered attributes. The first two classes are considered in this section, and the first and third classes will be discussed in next chapter.

3.6.1 Adjacency Matrix as Covariance Matrix

Consider a finite graph without self-loops $G(E, V)$ on n vertices $V = \{v_1, \cdots, v_n\}$. Then the *adjacency matrix* of graph G is an $n \times n$ matrix $A = [a_{ij}]$, where

$$a_{ij} = \begin{cases} \text{number of edges joining vertex } i \text{ and } j & \text{if } i \neq j \\ 0 & i = j \end{cases}.$$

Then, adjacency matrix A is symmetric but not necessary positive definite. For weighted graphs, a_{ij} should be the weight between node i and j. Let us consider the following matrix $K = [k_{ij}]_{n\times n}$, obtained by replacing a_{ii} to some positive number d. That is,

$$k_{ij} = \begin{cases} a_{ij} & i \neq j \\ d & i = j \end{cases} .$$

Since the diagonal elements of A are always zeros, K, called a *modified adjacency matrix*, contains all the information A has. By the following result [44], K is a positive definite matrix for a big enough d.

Theorem 2 *If an $n \times n$ symmetric matrix $A = [a_{ij}]$ is strictly diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{j=1, i\neq j}^{n} |a_{ij}| \quad \textit{for all } i = 1, \cdots, n. \tag{3.30}$$

Then, all the eigenvalues of A are real and positive, i.e., A is a symmetric definite matrix.

If graph G is an abstract representation of some system, a_{ij} represents the level of interactions between vertices i and j. The nodes can be viewed as random variables, and a_{ij} can be viewed as the covariance between random variable i corresponding to node i and random variable j corresponding to node j. Then, clustering and modularity of a graph G with modified adjacency matrix K will be equivalent to the clustering and modularity of a system of multivariate normal random variables with K as covariance matrix.

Example: As an example, let us consider the graph shown in Figure 3.14. The graph intuitively may have four clusters $\{\{1,2,3,4\}\{5,6,7,8\}\{9,10,11,12\}\{13,14,15,16\}\}$. Now let's assume the vertices are clustered into two subsets: $S_1(i) = \{1, \cdot, i\}$ and $S_2(i) = \{i+1, \cdots, 16\}, 2 \leq i \leq 14$. With i increasing from 2 to 14, M^1 and M^2 are calculated for each clustering, and the results are shown in Figure 3.15. The maxima along the curve give you the four clusters. By comparing Figure 3.15(a) and Figure 3.15(b), it is shown that the results of M^2 and M^1 are nearly the same.

Effects of Diagonal Element: When an adjacency matrix is modified into a covariance matrix, the diagonal element d could be any large enough positive number. The immediate question is how the diagonal element d affects the results. Figure 3.15(a)(b) shows that M^1 and M^2 are very robust on d. The intuitive explanation is that modularity measures the couplings between two modules, while d represents the variance of a random variables itself.

Effects of Vertex Ordering on M^1 and M^2: Another question worth of study is how the results are affected by the vertex ordering. Let us reorder the clusters and consider the following vertex order:$\{\{1,2,3,4\}\ \{13,14,15,16\}\ \{9,10,11,12\}\ \{5,6,7,8\}\}$, denoted as order 2. Let's denote the original order as order 1. In Figure 3.16(a)(b), the distributions of maxima of M^1 and M^2 do

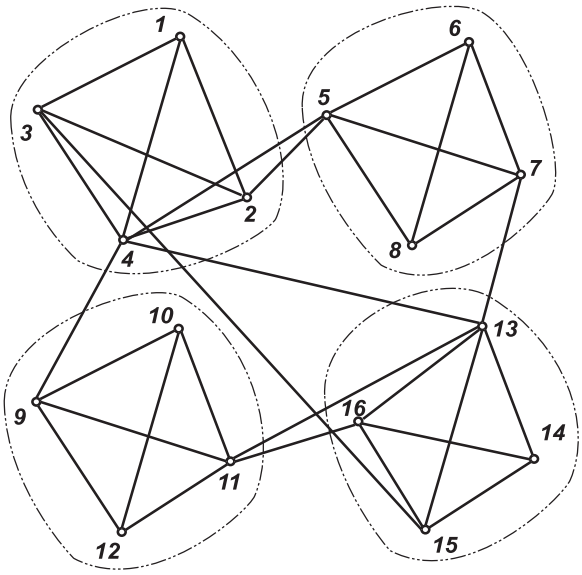


Figure 3.14: An example with 16 vertices.

not change, i.e., M^1 and M^2 are robust on the order of clusters. This will be verified more in Section 5.4, where all possible permutations will be considered.

■

3.6.2 Asymptotic Approximation as $d \longrightarrow \infty$

Now let us consider the asymptotic behaviors of M^1 and M^2 as $d \longrightarrow \infty$ under two-way clustering. The set of nodes is assumed to be split into two subsets, S_1 and S_2. Without loss of generality, it can be assumed that $S_1 = \{1, 2, \cdots, m\}$ and $S_2 = \{m+1, m+2, \cdots, n\}$. K is symmetric positive definite and can be represented as

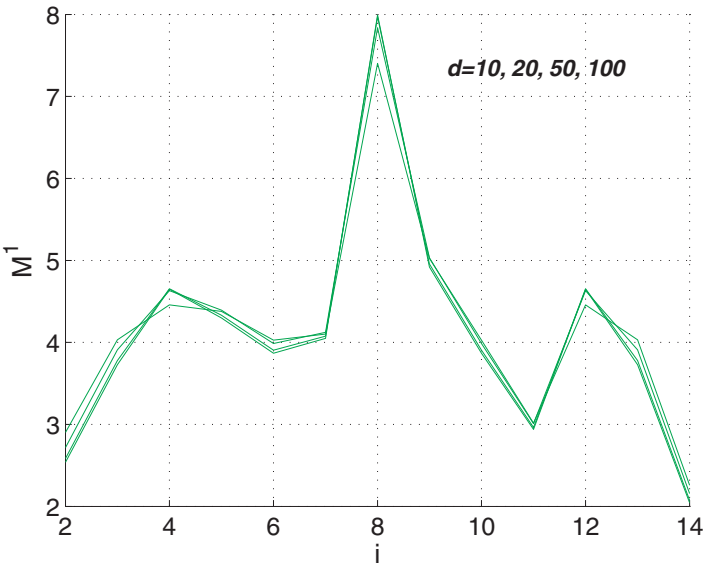
$$K = \begin{bmatrix} K_1 & B \\ B^T & K_2 \end{bmatrix}.$$

By the Leibniz formula,

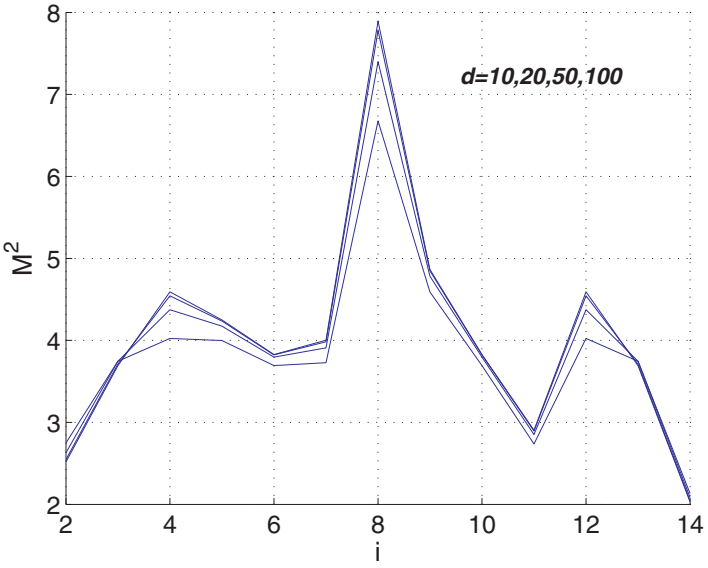
$$|K| = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} k_{i\sigma(i)},$$

where the sum is computed over all possible permutations σ of the numbers $\{1, 2, \cdots, n\}$, and $sgn(\sigma)$ denotes the signature of permutation σ, which is defined as

$$sgn(\sigma) = \begin{cases} 1 & \text{if } \sigma \text{ is an even permutation} \\ -1 & \text{if } \sigma \text{ is an odd permutation} \end{cases},$$

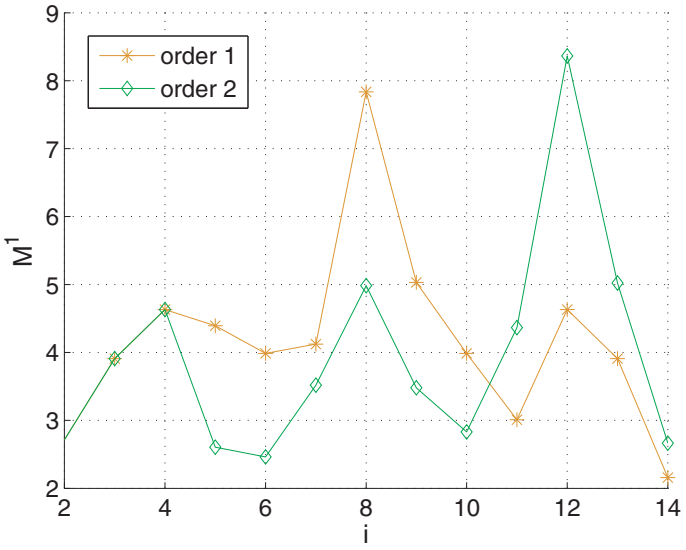


(a)

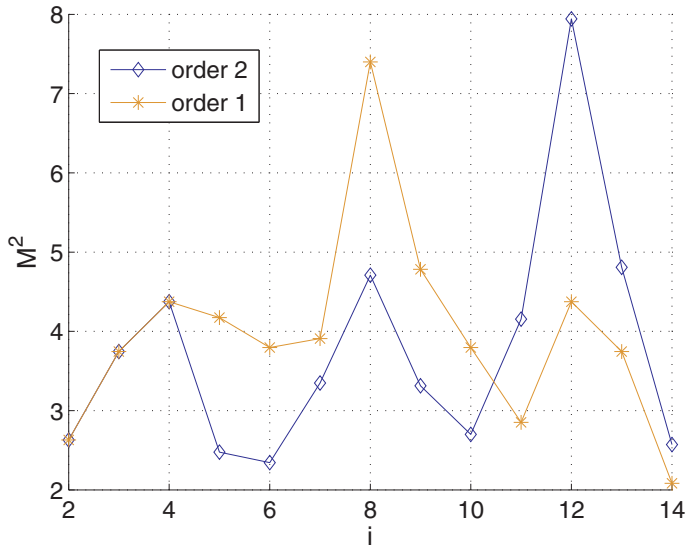


(b)

Figure 3.15: Effects of the diagonal element d on M^1, M^2.



(a)



(b)

Figure 3.16: Effects of vertex ordering on M^1, M^2.

where an even permutation is one that can be produced by an even number of exchanges of two elements, called *transpositions*, from permutation $(1, 2, \cdots, n)$, and an odd permutation is one that can be produced by an odd number of transpositions. Let us consider the coefficients of d^n, d^{n-1}, d^{n-2} terms in the expansion of $|K|$. Since ds are only on the diagonal, only those permutations, which exactly change i elements, can have d^{n-i} term in $\prod_{i=1}^{n} k_{i\sigma(i)}$. There is only one permutation, $(1, 2, \cdots, n)$, which doesn't change elements, and no permutation which only changes one element, so the coefficients of d^n and d^{n-1} are 1 and 0 respectively. Only transpositions can change two elements. Any transposition has form $(1, \cdots, i-1, j, i+1, \cdots, j-1, i, j+1, \cdots, n)$. That is, $\sigma(m) = m$ if $m \neq i, j$, $\sigma(i) = j$ and $\sigma(j) = i$. Note that $sgn(\text{transposition}) = -1$ and $k_{ii} = d$, the d^{n-2} term is

$$\sum_{\sigma \text{is transposition}} sgn(\sigma) \prod_{i=1}^{n} k_{i\sigma(i)} = -\sum_{i=1}^{n} \sum_{j=i+1}^{n} k_{ij} k_{ji} d^{n-2}.$$

Then

$$\begin{aligned}
|K| &= \sum_{\sigma \in s_n} sgn\,(\sigma) \prod_{i=1}^{n} k_{i,\sigma(i)} \\
&= d^n - \sum_{i=1}^{n} \sum_{j=i+1}^{n} k_{ij} k_{ji} d^{n-2} + o\left(d^{n-2}\right) \\
&= d^n \left(1 - \sum_{i=1}^{n} \sum_{j=i+1}^{n} k_{ij}^2 d^{-2} + o\left(d^{-2}\right)\right) \\
&= d^n \left(1 - \Sigma d^{-2} + o\left(d^{-2}\right)\right),
\end{aligned}$$

where $\Sigma = \sum_{i=1}^{n} \sum_{j=i+1}^{n} k_{ij}^2$. Then by Taylor expansion, $\ln(1-x) = x + o(x)$ for $x \ll 1$:

$$\ln |K| = n \ln d - \Sigma d^{-2} + o(d^{-2}).$$

Similarly,

$$\begin{aligned}
|K_1| &= d^m \left(1 - \sum_{i=1}^{m} \sum_{j=i+1}^{m} k_{ij}^2 d^{-2} + o\left(d^{-2}\right)\right) \\
&= d^m \left(1 - \Sigma_{11} d^{-2} + o\left(d^{-2}\right)\right) \\
\ln |K_1| &= m \ln d - \Sigma_{11} d^{-2} + o\left(d^{-2}\right),
\end{aligned}$$

where $\Sigma_{11} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} k_{ij}^2$, and

$$\begin{aligned} |K_2| &= d^{n-m} \left(1 - \sum_{i=m+1}^{n} \sum_{j=i+1}^{n} k_{ij}^2 d^{-2} + o\left(d^{-2}\right) \right) \\ &= d^{n-m} \left(1 - \Sigma_{22} d^{-2} + o\left(d^{-2}\right) \right) \\ \ln |K_2| &= (n-m) \ln d - \Sigma_{22} d^{-2} + o(d^{-2}), \end{aligned}$$

where $\Sigma_{22} = \sum_{i=m+1}^{n} \sum_{j=i+1}^{n} k_{ij}^2$. Then, by Equation 3.12

$$\begin{aligned} I(S_1 : S_2) &= \frac{1}{2}(\ln |K_1| + \ln |K_2| - \ln |K|) \\ &= \frac{1}{2} \left[m \ln d - \Sigma_{11} d^{-2} + (n-m) \ln d - \Sigma_{22} d^{-2} - n \ln d + \Sigma d^{-2} + o(d^{-2}) \right] \\ &= \frac{1}{2} (\Sigma - \Sigma_{11} - \Sigma_{22}) d^{-2} + o\left(d^{-2}\right) \\ &= \frac{1}{2} \Sigma_{12} d^{-2} + o(d^{-2}), \end{aligned}$$

where $\Sigma_{12} = \Sigma - \Sigma_{11} - \Sigma_{22} = \sum_{i=1}^{m} \sum_{j=m+1}^{n} k_{ij}^2$, and

$$I(X_i : X_j) = \frac{1}{2} \ln \frac{d^2}{d^2 - k_{ij}^2} = \frac{k_{ij}^2}{2d^2} + o(d^{-2}).$$

Then,

$$\begin{aligned} C_1^{in} &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \left(\frac{1}{2d^2} k_{ij}^2 + o(d^{-2}) \right) \\ &= \frac{1}{2} \Sigma_{11} d^{-2} + o(d^{-2}), \end{aligned}$$

and

$$\begin{aligned} I(S_1) &= \sum_{i=1}^{m} h(X_i) - h(X_1, \cdots, X_M) \\ &= -\frac{1}{2} \ln \frac{|K_1|}{d^m} \\ &= -\frac{1}{2} \ln (1 - \Sigma_{11} d^{-2} + o(d^{-2})) \\ &= \frac{1}{2} \Sigma_{11} d^{-2} + o(d^{-2}). \end{aligned}$$

From the above equations, C_1^{in} and $I(S_1)$ have the same asymptotic approximation to first order. Now let's consider the coupling

$$\begin{aligned} C &= \frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)} \\ &= \frac{\Sigma_{12}d^{-2} + o(d^{-2})}{m * (n-m)} \times \frac{m(m-1)}{2(\Sigma_{11}d^{-2} + o(d^{-2}))} \\ &\approx \frac{m(m-1)}{2m * (n-m)} \times \frac{\Sigma_{12}}{\Sigma_{11}}. \end{aligned} \tag{3.31}$$

For adjacency matrix, $k_{ij} = 1$, then

$$\begin{aligned} \Sigma_{11} &= \sum_{i=1}^{m}\sum_{j=i+1}^{m} k_{ij}^2 = \sum_{i=1}^{m}\sum_{j=i+1}^{m} 1 \\ &= \text{the number of links inside } S_1. \end{aligned}$$

Similarly, Σ_{22} is the number of links inside S_2, and Σ_{12} is the number of links between S_1 and S_2. Then, the second fraction in Equation 3.31 is

$$\frac{\Sigma_{12}}{\Sigma_{11}} = \frac{\text{the number of links between } S_1 \text{ and } S_2}{\text{the number of links inside } S_1},$$

and the coupling is

$$C = \frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)} \approx \frac{2\Sigma_{12}/m(m-1)}{\Sigma_{11}/m * (n-m)} = \frac{\text{the average number of links between } S_1 \text{ and } S_2}{\text{the average number of links inside } S_1}.$$

The mutual information-based modularity of the two-way clustering is

$$\begin{aligned} C^2 &= \frac{1}{2}\left(\frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_1)} + \frac{\bar{I}(S_1 : S_2)}{\bar{I}(S_2)}\right) \\ &= \frac{m-1}{4(n-m)}\frac{\Sigma_{12}}{\sum_{11}} + \frac{n-m-1}{4m}\frac{\Sigma_{12}}{\Sigma_{22}}. \end{aligned}$$

Since C_1^{in} and $I(S_1)$ (C_2^{in} and $I(S_2)$) have same approximation to the first order, C^1 equals C^2 up to the first order approximation.

3.7 Summary

In Chapter 2, several questions on modularity have been proposed: system vs. model, specific meanings of "inter-module" and "intra-module," relativity of couplings, aggregation of modularity at different level in a hierarchical structure, application domain of definition, and how to quantify couplings. This chapter has tried to discuss the last but most important question, which is the key

for quantitative measure of modularity.

In this chapter, mutual information-based methods have been proposed as a way to quantify interactions (couplings) and a framework to quantify modularity by mutual information has been established. Those methods are based on information flow instead of directly on the real strength of interactions between subsystems. The mutual information-based method has been discussed for simple systems of random variables, where interactions are symmetric and non-directional, and real complex stochastic systems including dynamic behaviors, design processes, manufacturing and other processes, where the interactions could be asymmetric and directional. Even some graph structures can be viewed as a covariance matrix of some gaussian random vector and therefore, modularity can be quantified by mutual information. It has been shown that the general linkage counting measure of modularity is a special case of the mutual information-based measure of modularity.

Chapter 4

MDL-Based Measure for Topology Modularity of Graphs

4.1 Introduction

In the previous chapter, mutual information-based measure has been proposed for the modularity of dynamic behaviors of final products, design processes, and other processes. At the early phase of design processes, designs are not finalized, and the physical behaviors and design parameters are not very clear. Yet function structures commonly exist at the early stages of design processes. So this chapter studies how to quantify the topology or structure modularity.

Function structures and design structure matrices (DSM), which general modular production design methods [29, 48, 63] are based on, can be represented by graph structures, and graph representations are more general. Besides as a modeling tool for function structures and DSM in engineering design, abstract graph representations are also used to model many complex systems [20, 55, 52], including bondgraphs [52] in the analysis of dynamical systems. This chapter proposed an information-theoretic method of modularity measure [107, 108]. It can be applied to graph structures whose edges have unordered attributes. The method is based on Shannon's measure of information complexity [88], specifically the minimal description length (MDL) principle, and built on the observation that there are strong (inverse) relations between complexity and modularity. Yassine et al. [113] introduce a modularity measure of design structure matrices (DSM) based on the minimal description length principle and use it to cluster (not hierarchically) DSM. This thesis work is independent of and more general than Yassine' work. It can be used to identify and measure modularity of graph structures and to decompose complex systems into hierarchical modular configurations.

4.2 Minimal Description Length Principle

The primary motivation for relying on Minimal Description Length (MDL) is the inductive inference of a general hypothesis for given data [77, 78, 79, 80, 104, 106, 105]. 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 a shorter message to describe the regularity than the length of the message needed to describe the data literally. So, 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 on 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 find a hypothesis which minimizes the approximate length. The practical applications of MDL use such approximations [105]. 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.

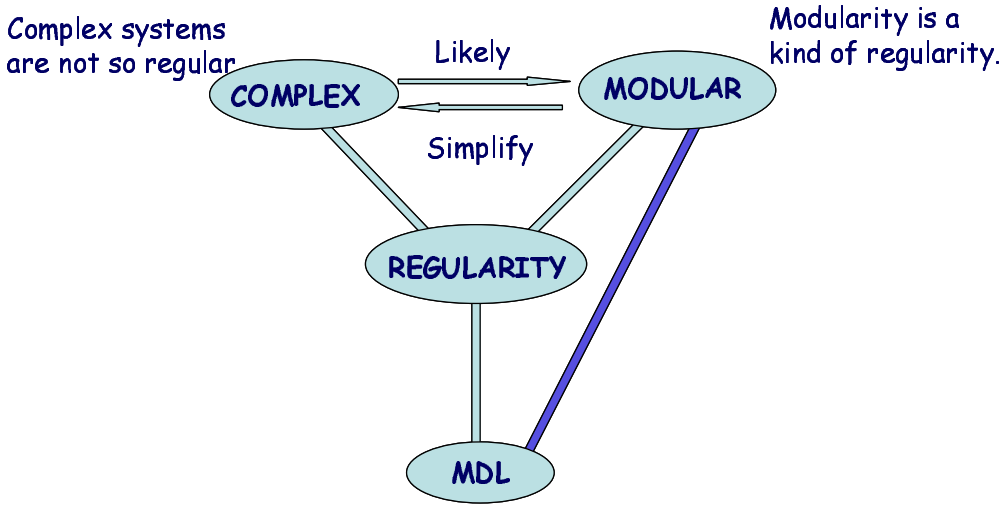


Figure 4.1: Modularity is a kind of regularity.

4.3 MDL and Modularity

Modularity is intrinsic and beneficial to complex systems [82] and may be a general principle for managing complexity. Modular architectures make complex systems easier to understand and manage and may play a critical role in the synthesis of complex artificial systems [102]. 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.

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. So, modularity can be viewed as a kind of regularity inside the complex systems, as shown in Figure 4.1. As discussed in the previous section, MDL is a kind of measure of regularity. It is natural to reason that MDL could provide some measure on modularity

In the framework developed in Chapter 2, the modularity of a system is defined as the maximum of the modularity of all feasible decompositions. With the MDL principle, different feasible hierarchical decompositions can be viewed as different *hypotheses* and the abstract graph as *data* waiting to be described by the decompositions. The higher the modularity of a system, the more compression can be made on the description length, and therefore the shorter description length. A measure of modularity of a decomposition can be considered to be the inverse of the minimal description length

of the messages describing a system under a decomposition.

Weak coupling between modules means that most of the units inside one module 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) are needed to describe (or encode) a modular system. Where a high level of intra-module 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 units inside modules 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 (shorter message) is required to describe a modular structure.

The information used to describe two systems realizing different customer requirements (or customer attributes) are totally different. The more functions a system realizes, the more information is needed to describe it. So, it is nonsense to compare the message lengths of two descriptions of two totally different complex systems, yet it is reasonable to compare normalized deductions of the message lengths. Suppose:

- L_0 is the description length of the complex system without decomposition.
- L_d is the message length of the complex system under some decomposition.

Then, the modularity of a decomposition is computed as

$$M_d = \frac{L_0 - L_d}{L_0}.$$

As shown in Figure 4.2(a), modular structures are hierarchical and represented as a tree structure, in which the whole system is represented as the root or trunk, and modules or submodules are represented as nodes or leaves. The modularity of a decomposition is an aggregation of the modularity of subsystems(modules) at different levels. Let's consider the hierarchical structure shown in Figure 4.2(a). There is only one subsystem R at level 1 (the highest level in the structure), which has two modules M_{21} and M_{22}. The tree structure of the module at level 1 is shown in Figure 4.2(b). The modularity at level 1 is considered to be how much compression can be caused by the modular structures at this level, and the measure is to compare the minimal description lengths at the following two cases. One case calculates the description length L_0^1 for the left tree structure shown in Figure 4.2(b). The other case considers the tree structure where there are no modules M_{21} and M_{22} as shown in the left tree structure in Figure 4.2(b), and calculates the description length L_d^1. Then, the modularity at level 1 is $M_d(1) = \frac{L_0^1 - L_d^1}{L_0^1}$.

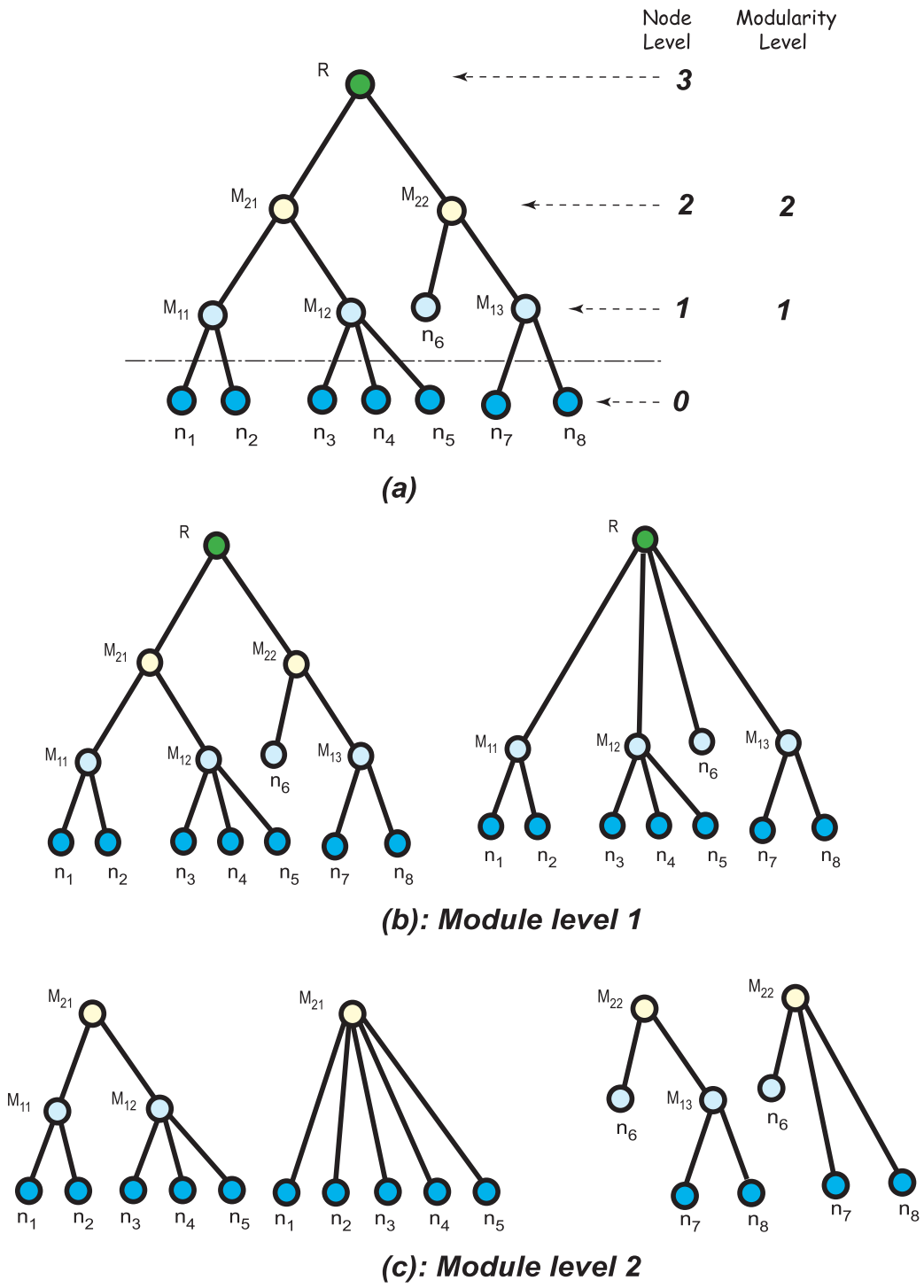


Figure 4.2: Modularity at different levels.

At level 2 there are two subsystems, M_{21} and M_{22}, as shown in Figure 4.2(c). The modularity at this level, $M_d(M_{21})$ and $M_d(M_{22})$, can be calculated with a method similar to that method used to calculate the modularity at level 1. Then, the modularity of level 2 is the average of $M_d(M_{21})$ and $M_d(M_{22})$. Then, the modularity of the decomposition is the aggregation of the modularity at different levels. The techniques discussed in Section 2.5.4 can still be used. A specific implementation on abstract graph structures $G(V, E)$ will be introduced in the next section.

4.4 Encoding Graph Structures

What is the information in a graph? Obviously it includes nodes, links, and attributed associated to links, and more it could include another thing called interfaces which are interpreted as the nodes of modules, which have links to nodes outside the modules. Interfaces do not exist in a graph without modular structure but in one with modular structure. In modular structures, interfaces clearly 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 ⟨SubMod M_1⟩⟨M_1's SubMod ⟩$\cdots$ ⟨M_i's SubMod ⟩⟨Interface in SubMod M_i⟩.

For the graph structure shown in Figure 4.3, nodes in the structure are labeled n_1 through n_{12}. Links represent interaction between different units. For instance, 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 labeled M_1^1 is a module, and the smaller regions labeled M_1^2 and M_2^2 are

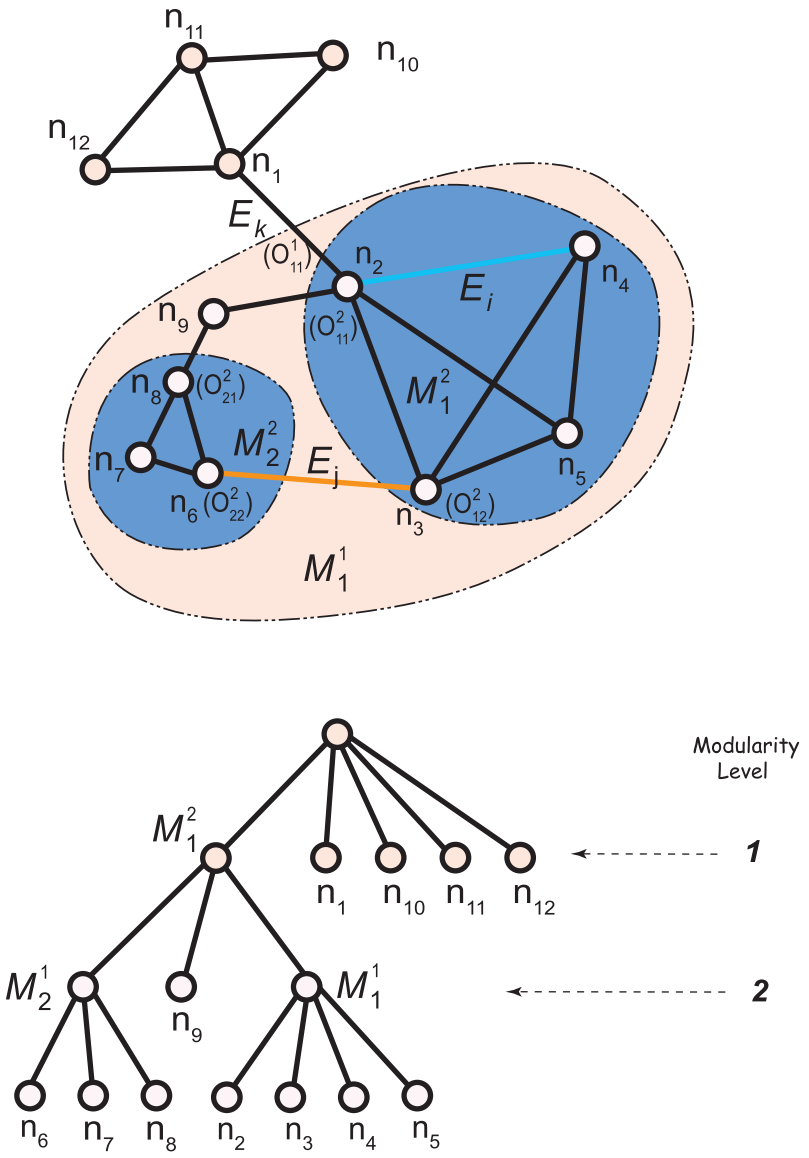


Figure 4.3: An example graph representation and its hierarchical tree representation.

submodules. Interface O^2_{11} is the node which is inside module M^2_1 but is visible to other nodes and links outside module M^2_1 in the sense that it can be connected to other nodes outside module M^2_1. Then the message format described above can be used describe the links and the graph structures. For instance, link E_i in Figure 4.3 can be represented as "$n_2 n_4\langle$attribute values$\rangle$." Nodes may be those inside submodules, e.g., link E_j in Figure 4.3 can be represented as "$M^2_2\ O^2_{22}$ $M^2_1\ O^2_{12}\langle$attribute values$\rangle$." Links can pass through several module levels. For example, the link E_k in Figure 4.3 passes through M^1_1 and into M^2_1. Then it can be represented as $n_1 M^1_1 M^2_1 O^2_{11}$.

An example message for the decomposition of the graph structure shown in Figure 4.3:

M^1_1 (Module name) O^1_{11} (List of Interfaces) {

M^2_1(Module name) $O^2_{11} O^2_{12}$(List of Interfaces) {

$n_2\ n_3$ (links)

$n_2\ n_4$

$n_2\ n_5$

$n_3\ n_4$

$n_3\ n_5$

$n_4\ n_5\}$

$M^2_2\ O^2_{21}\ O^2_{22}$ {

$n_6\ n_7$

$n_6\ n_8$

$n_7\ n_8\}$

$n_9\ M^2_1\ O^2_{11}$

$n_9\ M^2_2\ O^2_{21}$

$M^2_2\ O^2_{22}\ M^2_1\ O^2_{12}\}$

$n_1\ n_{10}$

$n_1\ n_{12}$

$n_{12}\ n_{11}$

$n_{11}\ n_{10}$

$n_1 M^1_1 M^2_1 O^2_{11}$.

4.4.1 Names and Links

Links can be represented as messages containing codes for two nodes and attributes. For instance, link L_i in Figure 4.3 can be represented as "$n_2 n_4 <$ attribute values $>$." The nodes may be those inside submodules, e.g., link L_j in Figure 4.3 can be represented as "$n_5 M_j n_6 <$ attribute values $>$." Messages for attributes will be discussed in Section 4.4.2. This section will discuss encoding for node

names. Now consider a module, and suppose that there are $N^{(n)}$ nodes and $N^{(m)}$ submodules. 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 $L_t^m + L_{tv}^{(o)} + L_s^{(m)} + L_{su}^{(o)}$.

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:

$$\begin{aligned} I^{(nk)} &= \text{Sum of the lengths of all links in the module} \\ &= \sum_{j=1}^{N^{(n)}} L_j^{(n)} \times N_j^{(n)} + \sum_{j=1}^{N^{(m)}} L_j^{(m)} \times N_j^{(m)} \\ &\quad + \sum_{j=1}^{N^{(m)}} (\sum_{k=1}^{N_j^{(o)}} L_{jk}^{(o)} \times N_{jk}^{(o)}), \end{aligned} \tag{4.1}$$

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 [88], 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:

$$-\sum_{j=1}^{N_i^{(u)}} N_j \log(p_j) = -\sum_{j=1}^{N_i^{(u)}} N_j \log(N_j / \sum_{j=1}^{N_i^{(u)}} N_j) \tag{4.2}$$

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

$$\begin{aligned} I_n^{(n)} &= \sum_{\text{all modules at level } n} I^{(nk)} \\ &= -\sum_{\text{all}} \sum_{j=1}^{N_i^{(u)}} N_j \log(N_j / \sum_{j=1}^{N_i^{(u)}} N_j) - \\ &\quad \sum_{\text{all}} \sum_{j=1}^{N^{(m)}} (\sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)} \log(N_{jk}^{(o)} / \sum_{k=1}^{N_j^{(o)}} N_{jk}^{(o)})) \end{aligned} \tag{4.3}$$

4.4.2 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, only unordered attributes are considered, mainly due to the following two reasons. One is that most common graph representations, such as function structures, only have unordered attributes, and the other is that there are no good encoding methods for continuous values. One possible way is to discretize a continuous value into an integer, then encode the integer. The encoding length heavily depends on the integer value, which is determined by the discretization step. However, there is no criterion to choose the discretization step.

For unordered attributes, 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

$$p_{t_j} = \frac{N_{t_j}}{\sum_{t_j} N_{t_j}}. \tag{4.4}$$

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

$$I^{(T_i)} = -\sum_{j=1}^{N_T} \sum_{t_j} N_{t_j} \log(p_{t_j}). \tag{4.5}$$

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

$$I_n^{(mA)} = -\sum_{\text{all modules at level } n} \sum_{j=1}^{N_T} \sum_{t_j} N_{t_j} \log(P_{t_j}). \tag{4.6}$$

4.4.3 Total Message Length

The total message length of level n is

$$I_n = I_n^{(n)} + I_n^{(mA)}. \tag{4.7}$$

Then, the overall description length is

$$I = \sum_{i=1}^{n} I_n = \sum_{i=1}^{n} I_n^{(n)} + I_n^{(mA)}. \tag{4.8}$$

Table 4.1: Code lengths for units in case without decomposition.

Node	# of Links	n_i/N	Code Length
n_1	2	1/9	$\log 9$
n_2	1	1/18	$\log 18$
n_3	4	2/9	$\log 9 - 1$
n_4	5	5/18	$\log 18 - \log 5$
n_5	3	1/6	$\log 6$
n_6	3	1/6	$\log 6$

4.4.4 Modularity Measure

The description length of any tree structure can be calculated by the above process. Then, the modularity M_i at a specific level i can be calculate the method discussed in Section 4.3. The modularity at different levels can be aggregated into the overall modularity M by some aggregation function f,

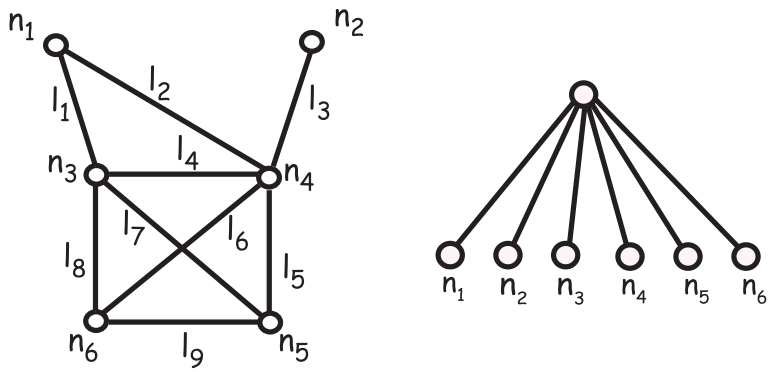
$$M = f(M_1, \cdots, M_n). \tag{4.9}$$

4.5 Examples

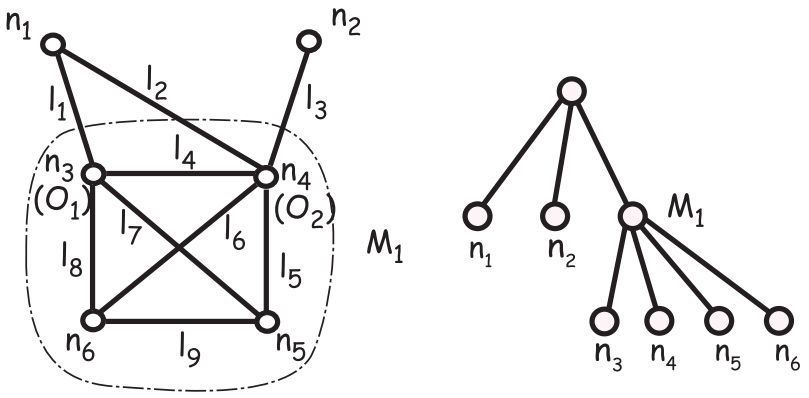
One simple example is shown in Figure 4.4(a). There are 6 nodes, whose coding lengths are shown in table 4.1. The message lengths of links are shown in table 4.2. Then, the total message length L_0 is 43.94 bits.

Now let us consider one possible decomposition (one level, not hierarchical) shown figure 4.4(b). There are three units, i.e., one module M_1 and two nodes n_1 and n_2, and inside module M_1 there are four nodes, n_3, n_4, n_5, and n_6, and two interfaces, O_1 and O_2. First, consider the message length for nodes inside module M_1. The code lengths for nodes and interfaces are shown, respectively, in Table 4.3 and Table 4.4. There are 6 links inside M_1, and they have all length $2+2=4$, so the total message length I_0 for links inside module M_1 is $6 \times 4 = 24$. Now consider the message for nodes and links at level 1. The code lengths for nodes and links are shown, respectively, in Table 4.5 and Table 4.6. The total message length for links at this level I_1 is 14.90. Then, the total description length of the system under the decomposition is $L_d = I_0 + I_1 = 38.90$ bits. So the modularity of the system under the decomposition is

$$M_d = \frac{L_0 - L_d}{L_0} = \frac{43.94 - 38.90}{43.94} = 0.115.$$



(a) Original graph and its hierarchy tree.



(b) One possible decomposition and its hierarchy tree.

Figure 4.4: A simple example for information measure of modularity of graph structures.

Table 4.2: Code lengths for units in case without decomposition.

Link #	representations	Coding Length
l_1	$n_1 n_3$	$2\log 9 - 1$
l_2	$n_1 n_4$	$\log 9 + \log 18 - \log 5$
l_3	$n_2 n_4$	$2\log 18 - \log 5$
l_4	$n_3 n_4$	$\log 18 + \log 9 - \log 5 - 1$
l_5	$n_4 n_5$	$\log 18 - \log 5 + \log 6$
l_6	$n_4 n_6$	$\log 18 - \log 5 + \log 6$
l_7	$n_3 n_5$	$\log 9 - 1 + \log 6$
l_8	$n_3 n_6$	$\log 9 - 1 + \log 6$
l_9	$n_5 n_6$	$2\log 6$
	Total	43.9392 bits

Table 4.3: Code lengths for units in M_1.

Node	# of Links	Code Length
n_3	3	$-\log(1/4) = 2$
n_4	3	2
n_5	3	2
n_6	3	2

Table 4.4: Code lengths for interfaces in M_1.

Interfaces	# of Links	Code Length
O_1	1	$-\log 1/3 = \log 3$
O_2	2	$-\log 2/3 = \log 1.5$

Table 4.5: 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 4.6: Code lengths for links at level 1.

Link	Representation	Code Length
1	$n_1 M_1 O_1$	$\log 3 + 1 + \log 3$
2	$n_1 M_1 O_2$	$\log 3 + 1 + \log 1.5$
3	$n_2 M_1 O_2$	$\log 6 + 1 + \log 1.5$
	Total:	14.90

4.6 Summary

Since structure modularity is commonly used in engineering practice, especially at the early phase of engineering design, an MDL-based measure of modularity is proposed for abstract graph structures. The idea is based on the observation that modularity can be thought as a kind of regularity which can be used to compress data and can be measured by MDL. Therefore, MDL can provide a measure of modularity under some situations. In the next chapter, the MDL-based measure will be used to hierarchically decompose graph structures and function structures in engineering design.

Chapter 5

Module Identification

5.1 Introduction

A mutual information-based modularity measure for random systems was developed in Chapter 3, and Chapter 4 has developed an MDL-based measure for graph representations with or without attributes. This chapter will demonstrate those two methods by using them to hierarchically decompose abstract graphs and real function structures in engineering design. It is common in engineering practice to hierarchically decompose large complex systems into small subsystems. A particularly interesting situation is in redesign processes, where a configuration already exists, and it is required to redesign the architectures of products to make them more efficient and more modular.

At the very beginning phase of design, there are only function structures and no physical realizations, and furthermore it's pointed out that it is better to produce modular structures as early as possible. So, it is significant to consider the modularization of function structures. Function structures can be abstracted as graph representations. 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 of graph representation is shown in Figure 5.1. In engineering modeling, it is common to associate the links of graphs with some attributes representing the strength and types(domains) of interactions. For example, the interactions in Figure 5.1 could be energy, signal, or material.

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 [25] 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 using a genetic algorithm, whose search procedure is based on the mechanism of natural selection.

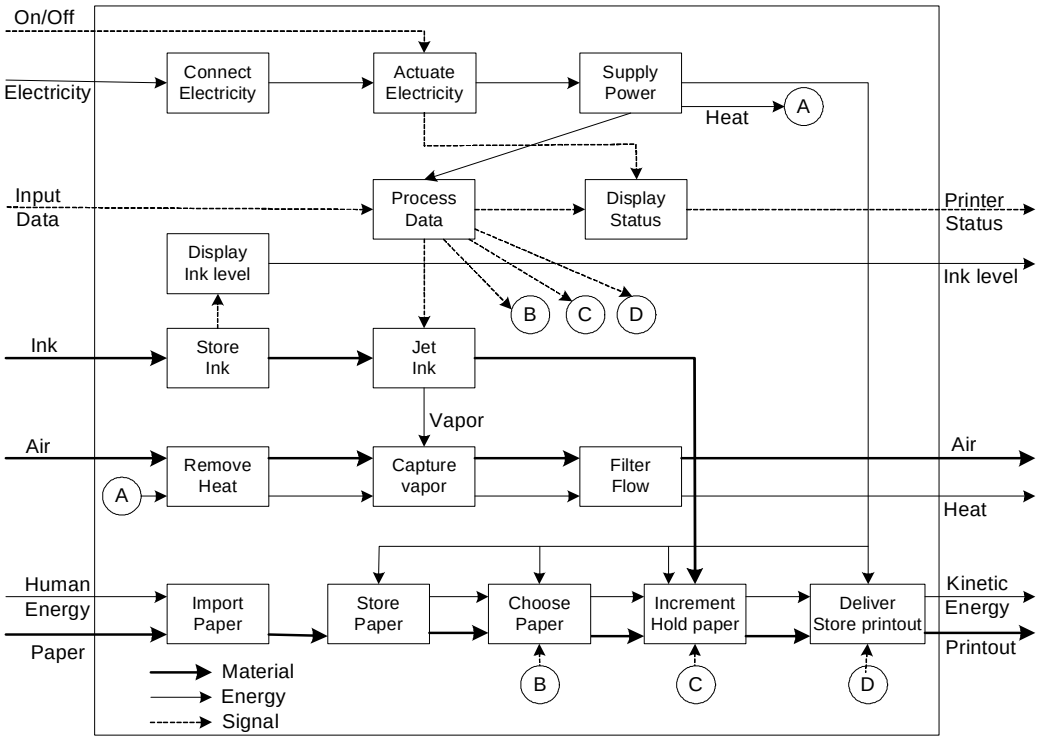


Figure 5.1: Graph representation of a function structure. It's a function structure of an HP 1200C desktop inkjet printer from K. Otto and K. Wood's Product Design [70]

5.2 Brief Introduction to Genetic Algorithm

Genetic algorithms were introduced in [41], and they were subsequently developed by Goldberg et al. [30]. The algorithms maintains a population whose individuals are encoded forms of solutions to the problem under consideration and iteratively evolve the population to the optimal or near-optimal solution. Each individual is called a genome, and each iteration is called a generation. At the beginning of the algorithm, the initial population is generated randomly. During each generation, the individuals (genomes) are evaluated by a fitness evaluation function and selected according to the fitness values using a selection mechanism so that fitter individuals (genomes) have higher probabilities of being selected. New individuals (genomes) can be formed by either exchanging genetic information between two individuals (genomes) selected from the current generation using a crossover operator or modifying an individual (genome) using a mutation operator. Crossover potentially leads to a better pool of population since it happens between two relatively better individuals of the population, while mutation is to ensure a more thorough exploration of the search space since it randomly introduces new features. Crossover and mutation is to balance exploitation and exploration. A new population is generated by selecting individuals (genomes) from the present population and the new generated individuals (genomes). Because of exploitation of crossover operators and exploration of mutation operators, the algorithm will converge to the optimal or near-optimal solution.

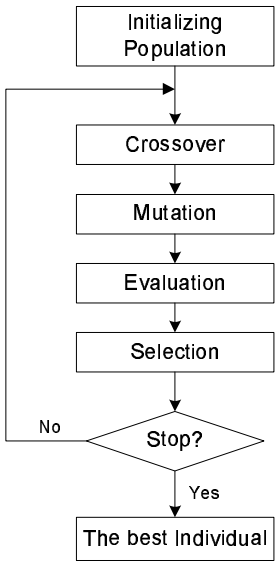


Figure 5.2: General structure of genetic algorithm.

A general structure of genetic algorithms is shown in Figure 5.2 in flowchart form. In order to use genetic algorithms, the following elements must be provided:

1. *Encoding schemes.* Since GAs work with a coding of the parameter set, not on parameters themselves, an encoding scheme is required. The scheme used in the following experiments is discussed in Section 5.3.1.

2. *Population initialization.* A commonly used way is to randomly and uniformly sample the search space, which is used in the following experiment.

3. *Genetic operators.* The population is evolved with mutation and crossover operators. The operators in this experiment are discussed further in Section 5.3.3.

4. *Evaluation Function.* A measure is used to tell how fit individuals are as solutions to the problem.

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 [30]. 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,

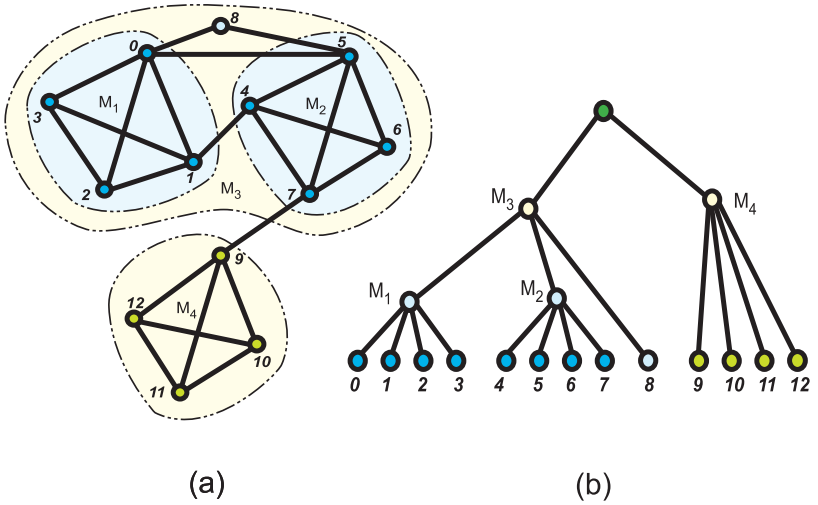


Figure 5.3: Tree representation of hierarchically modular structures.

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 the weakest individuals will be replaced by offsprings.

5.3 Computation: Algorithm Setup

In the following experiments, steady state genetic algorithms are used, and 12% individuals are replaced in every generation. Roulette wheel selection is used to select parents, and deterministic selection is used for generational selection. The encoding scheme, fitness function, and genetic operators are discussed in the following sections.

5.3.1 Encoding Scheme

As shown above, decomposition is to cluster the nodes of a graph. The only thing that needs to be encoded is which cluster a node belongs to. It's unnecessary to encode links in order to represent a decomposition. 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 one graph node. For example, the modular structure shown in Figure 5.3(a) could be represented as shown in Figure 5.3(b),

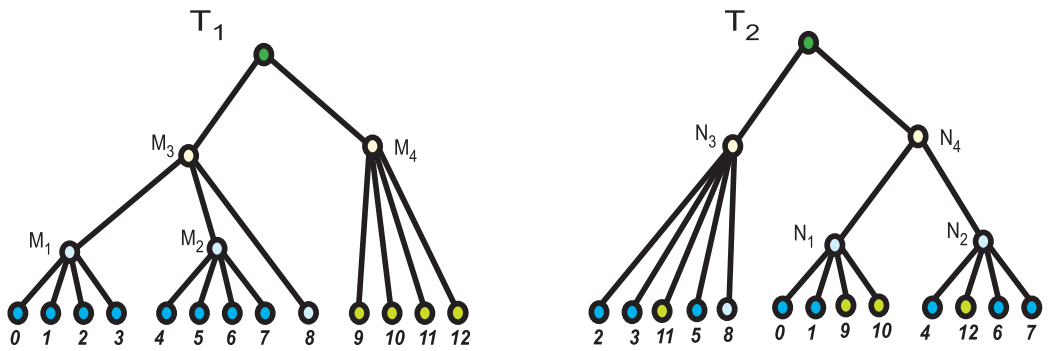


Figure 5.4: Crossover step 1: genome structures of initial parents.

5.3.2 Fitness Function

The information-theoretic measure of modular structures is used as an indicator of fitness. The fitness of genomes x is defined to be:

$$\text{The Fitness } f(x) = M(x), \tag{5.1}$$

where $M(x)$ is the modularity measure of the modular structure encoded as genome x.

5.3.3 Genetic Operators

There are some existing operators for tree structures [56]. But those methods can not guarantee that the resulting tree structures are legal in the sense that every leaf of the tree represents one graph node, and every graph node appears once on the leaves of the tree representation. There are two approaches 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. It is not very clear how to integrate penalty into the fitness function in the first method, so the later method is used here. That is, it is necessary to design new genetic operators.

5.3.3.1 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 5.4.

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 5.5.

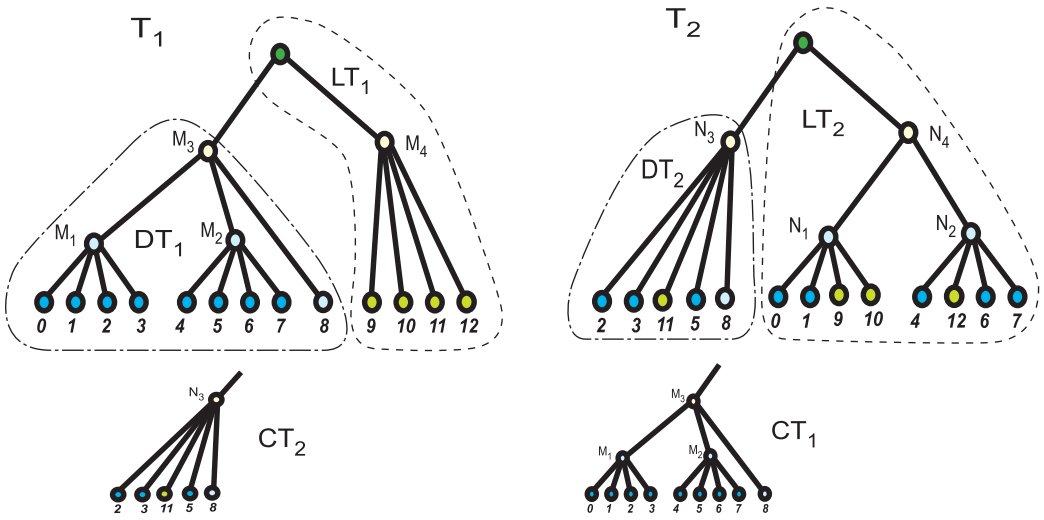


Figure 5.5: Crossover step 2: select crossover points.

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 5.6.

4. Randomly select a hidden node from (T_1 - LT_1) and add subtree CT_2. Do the same for T_2 and CT_1, as shown in Figure 5.7.

5. Clean fragments. 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 5.8.

5.3.3.2 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 5.9.

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 5.10.

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 5.11.

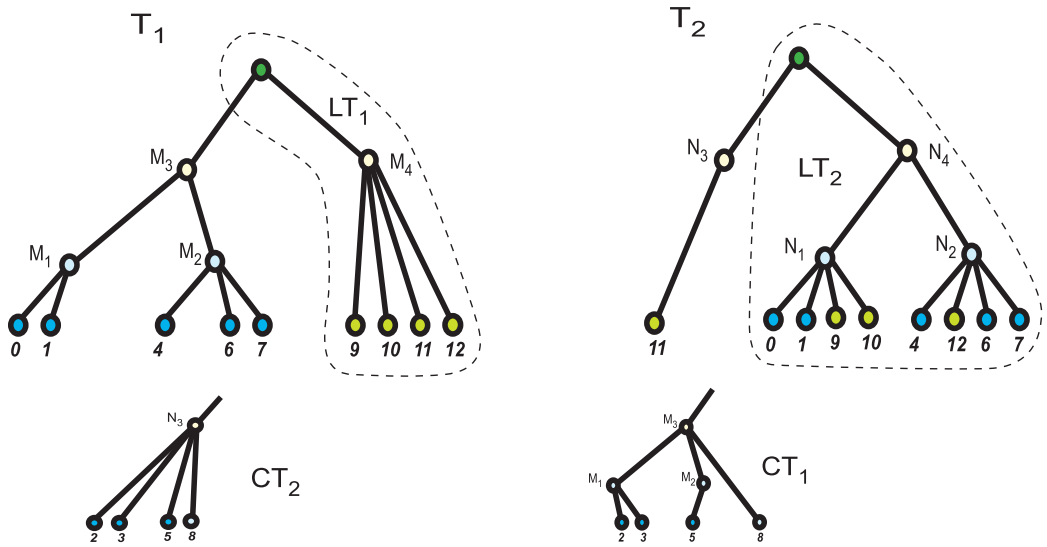


Figure 5.6: Crossover step 3: delete repeated leaves.

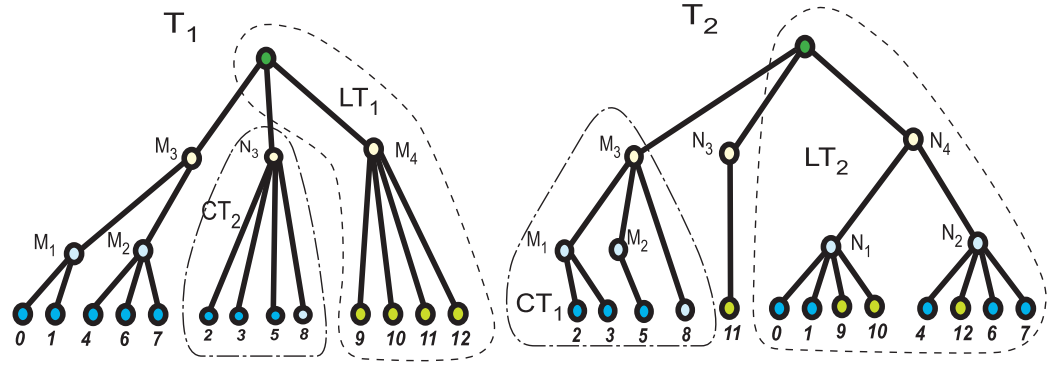


Figure 5.7: Crossover step 4: add subtrees.

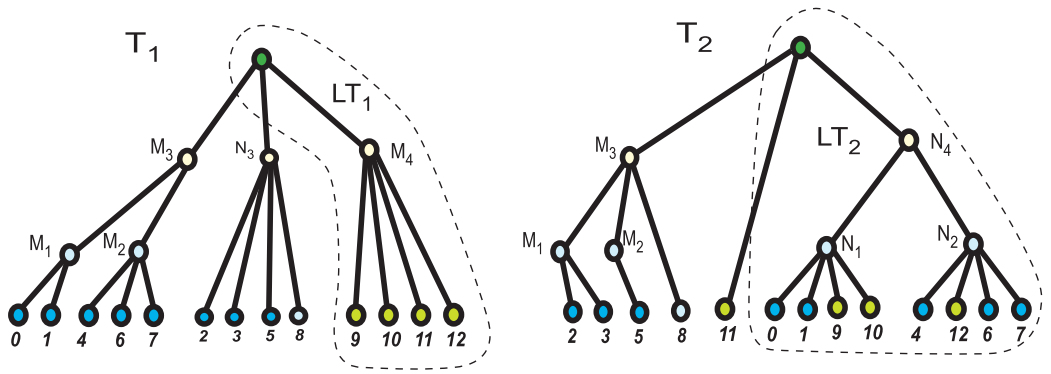


Figure 5.8: Crossover step 5: clean fragments.

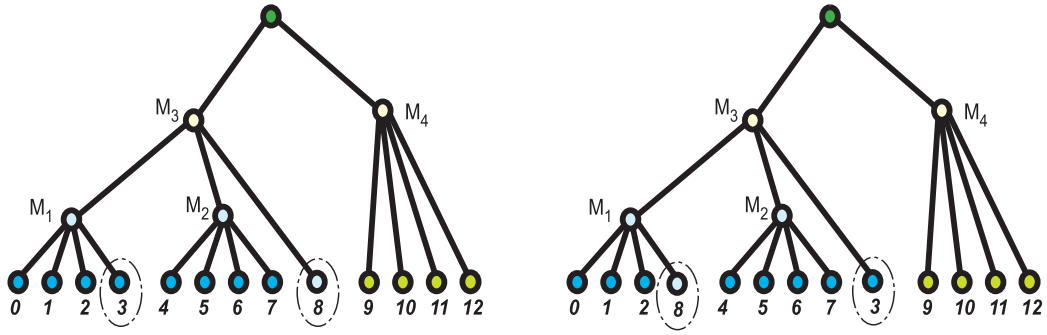


Figure 5.9: Mutation: swapping two leaves.

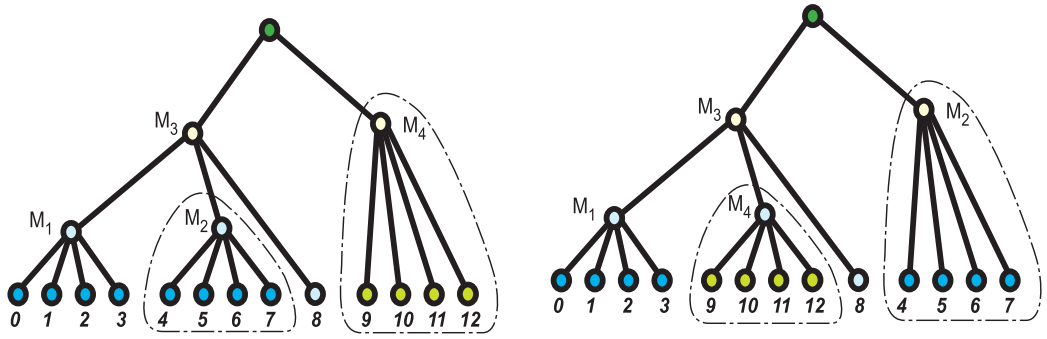


Figure 5.10: Mutation: swapping two subtrees.

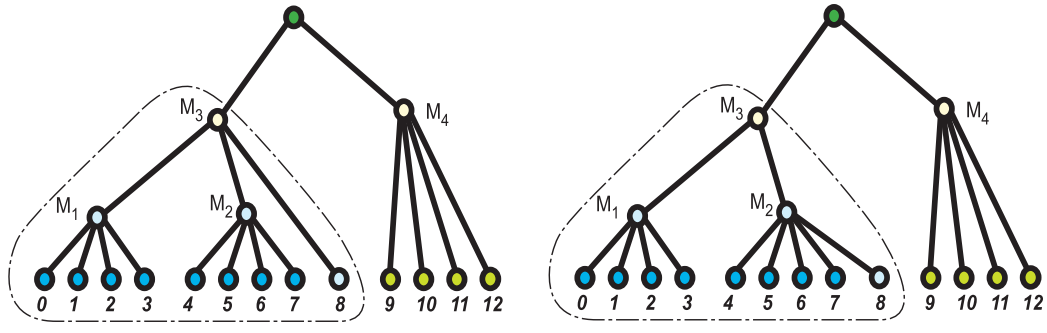


Figure 5.11: Mutation: merging single nodes into other subtrees.

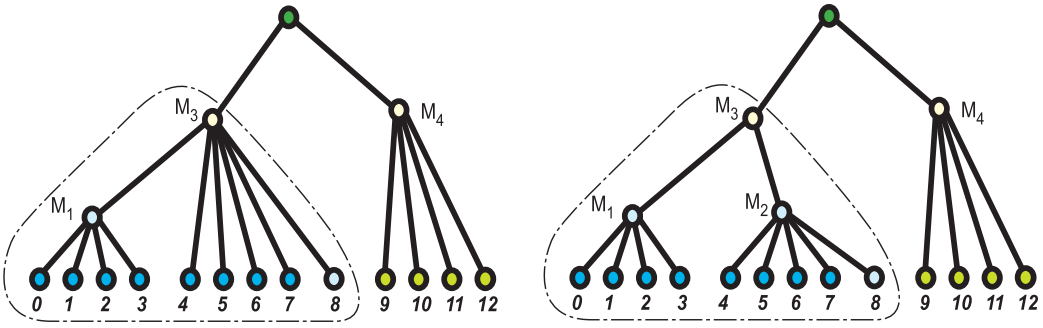


Figure 5.12: Mutation: merging single nodes as a subtree.

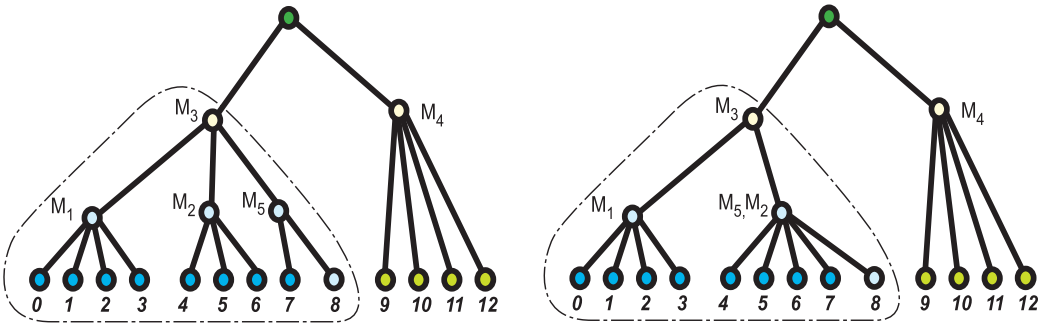


Figure 5.13: Mutation: merging two subtrees.

- *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 5.12.
- *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 5.13.

4. *Splitting a large subtree:* If a tree has many subtrees, it will be split into two subtrees, as shown in Figure 5.14.

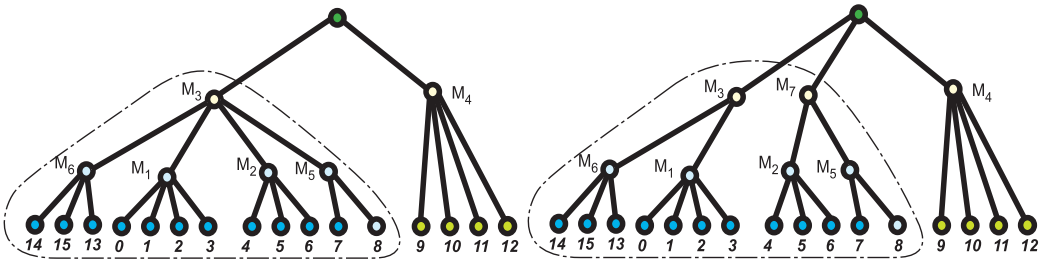


Figure 5.14: Mutation: splitting a subtree.

5.4 Abstract Graph Without Attributes

As discussed in Chapter 4, the MDL-based measure can be used to measure the modularity of abstract graph structures without ordered attributes. As for the mutual information-based modularity measure, it can also be used to decompose an abstract graph without unordered attributes if the adjacency matrix of the graph is viewed as a covariance matrix in some way, as discussed in Section 3.6. So in this section, both measures are verified on decomposition of abstract graphs without attributes.

5.4.1 Results and Discussions

In this experiment, the mutual information-based measure and MDL-based measures are used to decompose two examples of abstract graphs. The decomposition procedure is as follows: Given an abstract graph without attributes, the nodes of the graph are decomposed into a modular structure, and the modularity of the decomposition is calculated. Then, the best decomposition of the abstract graph is searched by evolutionary computation. In the genetic algorithm, the crossover rate is set to 0.9 and the mutation rate to 0.3. The initial population is set to 50 and randomly initialized, and 50 offspring are generated by crossover and mutation operators. The worst 6 individuals in parent generation are replaced by the best 6 offspring. This process is repeated until the population converges.

The first example is shown in Figure 5.15(a), which has a flat, not hierarchical, structure. In the mutual information-based method, the diagonal element of covariance matrix (modified adjacency matrix) is set to d when changing adjacency matrix into covariance matrix. Since the maximal degree of vertices is 5, d takes values $10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, \infty$. The $d = \infty$ case corresponds to general linkage counting methods. The mutual information-based method gives you the same decomposition under different ds, shown in Figure 5.15(b). The convergence of evolutionary computation with $d = 10$ is shown in Figure 5.16.

The MDL-based method also gives you the same decomposition, and the convergence of evolutionary computation is shown in Figure 5.18.

Another example is shown in Figure 5.19(a), which has a hierarchical structure. In mutual information based method, the diagonal element of covariance matrix (modified adjacency matrix) is also set to be d, which takes values $10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, \infty$. In this case, we need to consider the aggregation of modularity measures at different levels. In this experiment, they are uniformly weighted. The mutual information-method with different ds gives you the same decomposition shown in Figure 5.19(b). The convergence of evolutionary computation with $d = 10$ case is shown in Figure 5.20.

The MDL-based method also gives the same decomposition, and the convergence of evolutionary

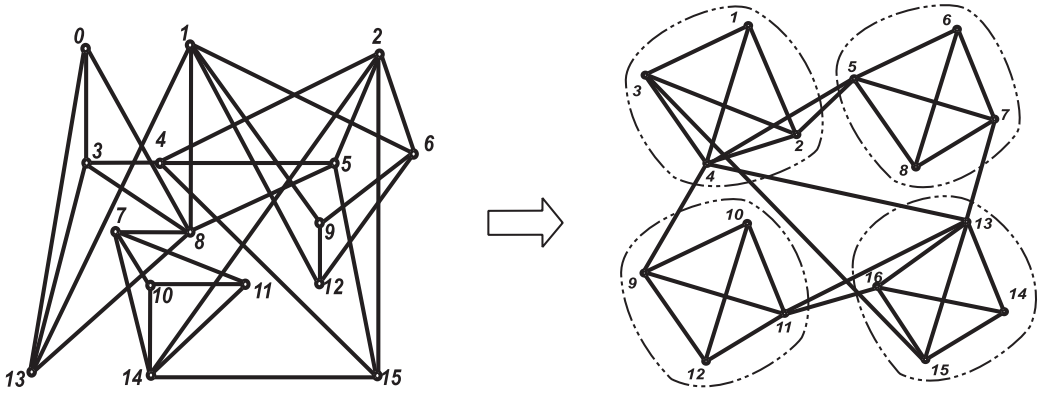


Figure 5.15: Decomposition result for example 1.

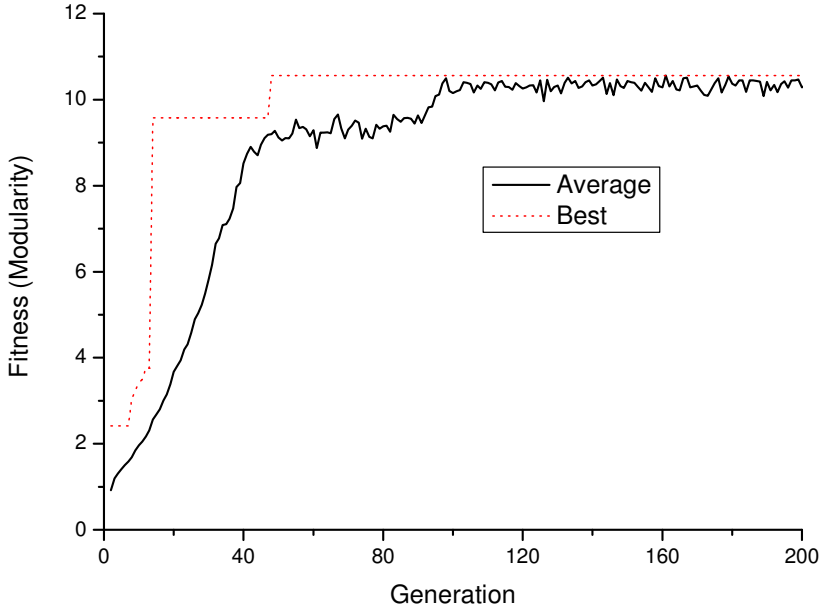


Figure 5.16: GA convergence of mutual information-based measure for example 1.

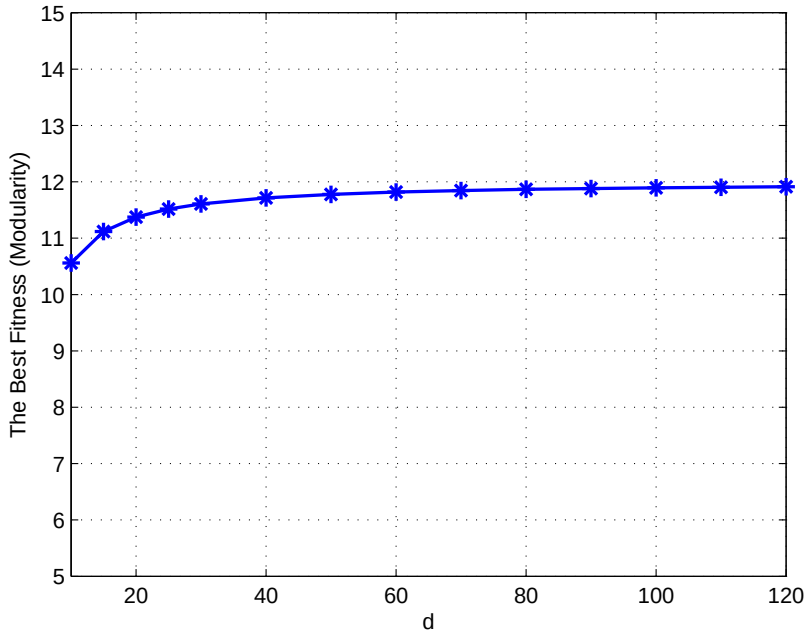


Figure 5.17: The effect of d on the fitness (modularity) of the best individuals in example 1.

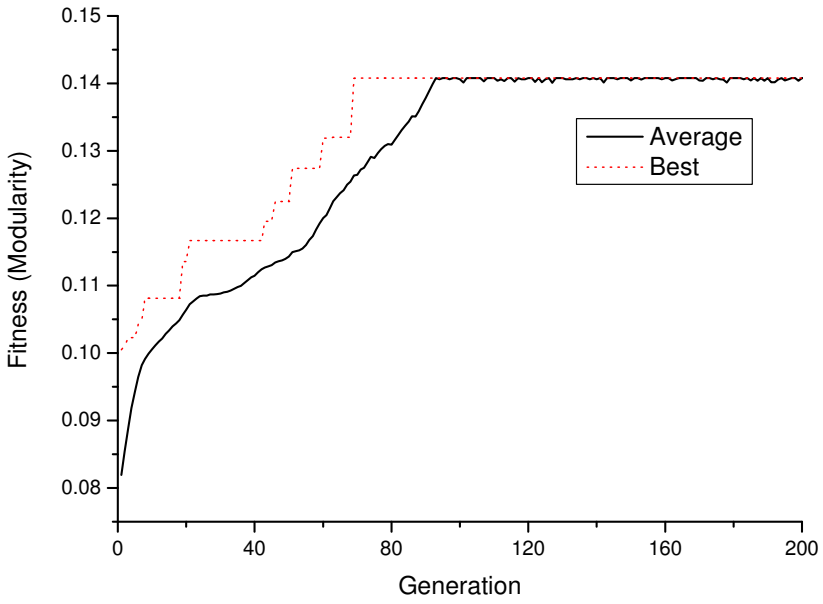


Figure 5.18: GA convergence of MDL-based measure for example 1.

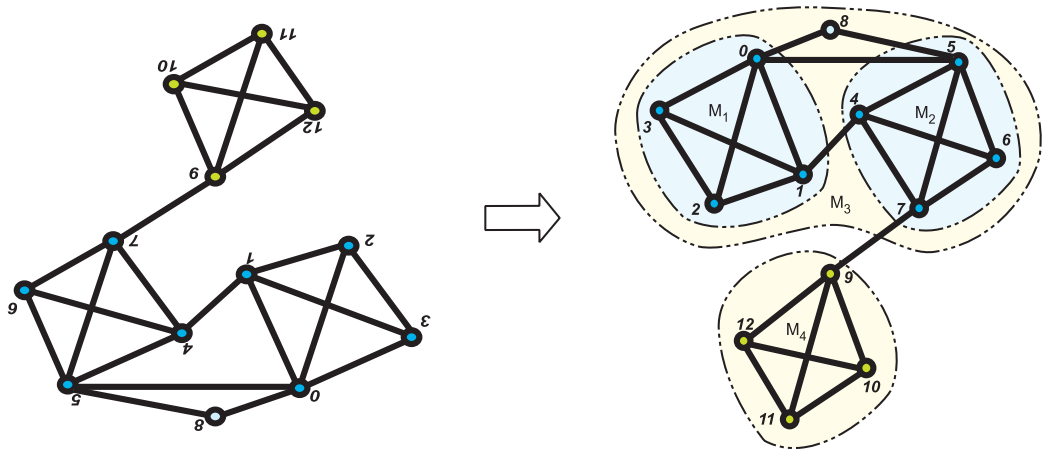


Figure 5.19: Decomposition result for example 2.

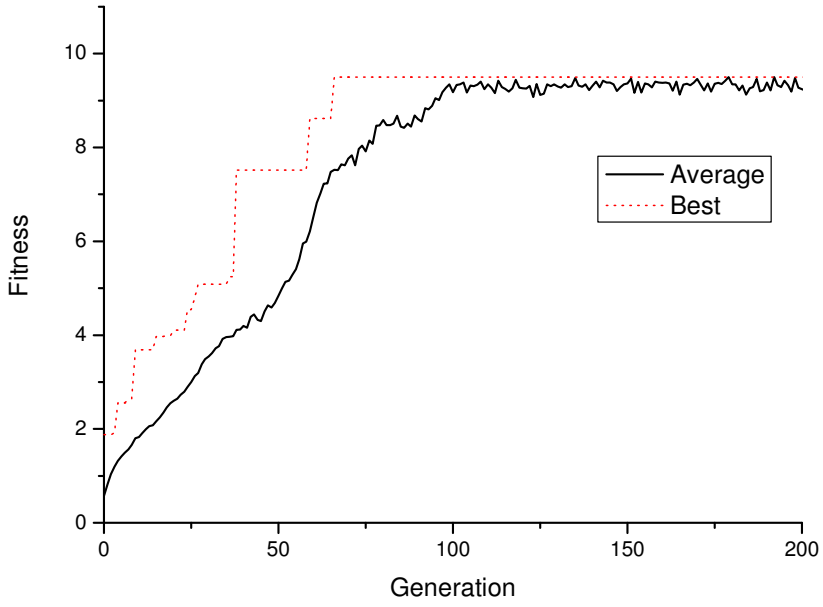


Figure 5.20: GA convergence of mutual information-based measure for example 2.

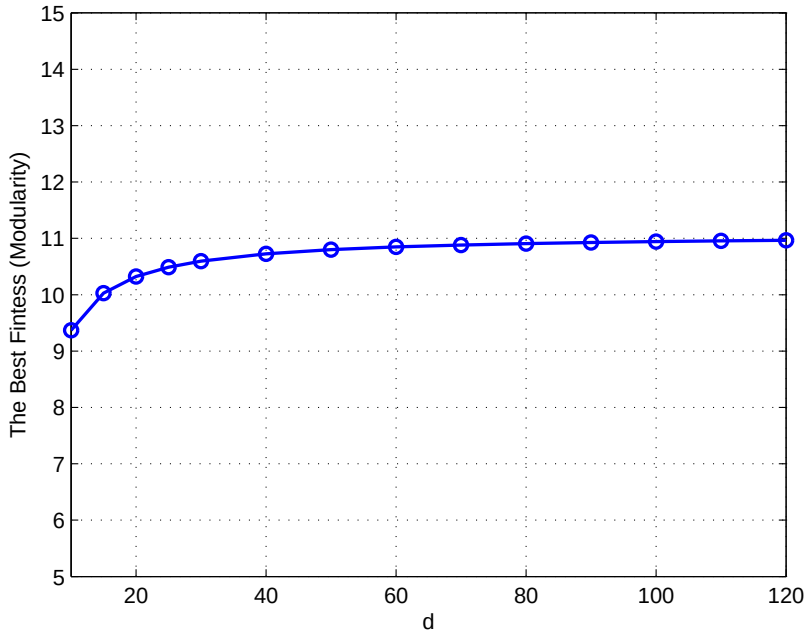


Figure 5.21: The effect of d on the fitness (modularity) of the best individuals in example 2.

computation is shown in Figure 5.22.

5.5 Function Structures

Function structures can be mapped into an abstract graph which can be accessed by computers. For example, the function structure of HP1200C printers [70]shown in Figure 5.1 can be presented as a graph shown in Figure 5.23(a). It is common that there are several attributes associated to the edges of an abstract graph, and those attributes are unordered. So, the mutual information-based measure is not applicable in this case. As pointed out in Chapter 4, the MDL-based method still works. In this experiment, evolutionary computation is used to decompose the function structure of HP1200C printers on the basis of MDL-based modularity measure.

5.5.1 Pre-measure

Generally, physical semantics of attributes associated to an abstract graph as well as engineering design practical experience are not shown in function structures and, therefore, neither in their abstract graph representations. How then is this knowledge and design expertise integrated into the decomposition process?

For decomposing the function structure of the HP printer, a pre-measure step is introduced, which is strongly related to the interfaces in modular structures. In engineering design, if one module M has

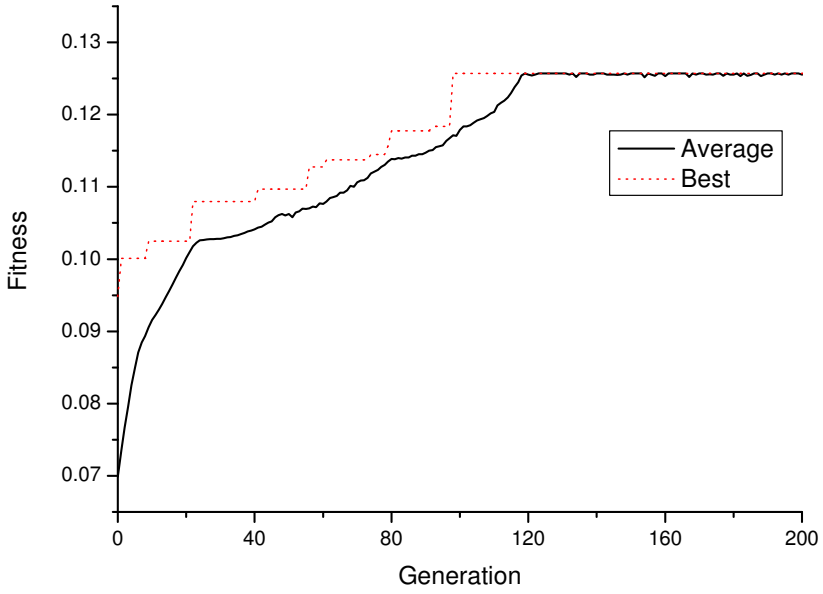


Figure 5.22: GA convergence of MDL-based measure for example 2.

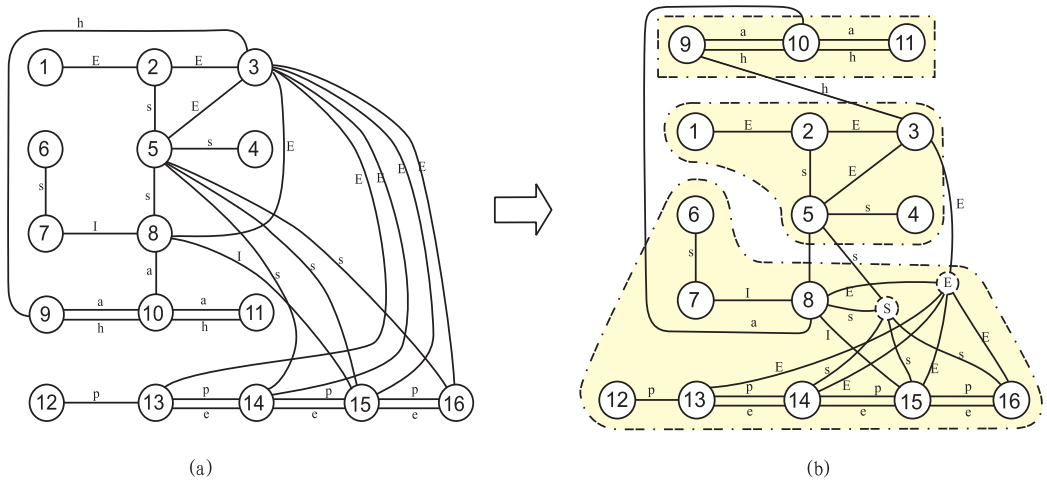


Figure 5.23: Function structure decomposition: (a) The abstract graph representation of the function structure of HP1200C. E: power; e: human energy; s: signal; p: paper; I: ink; h: heat; a: air. (b) One decomposition found by the algorithm.

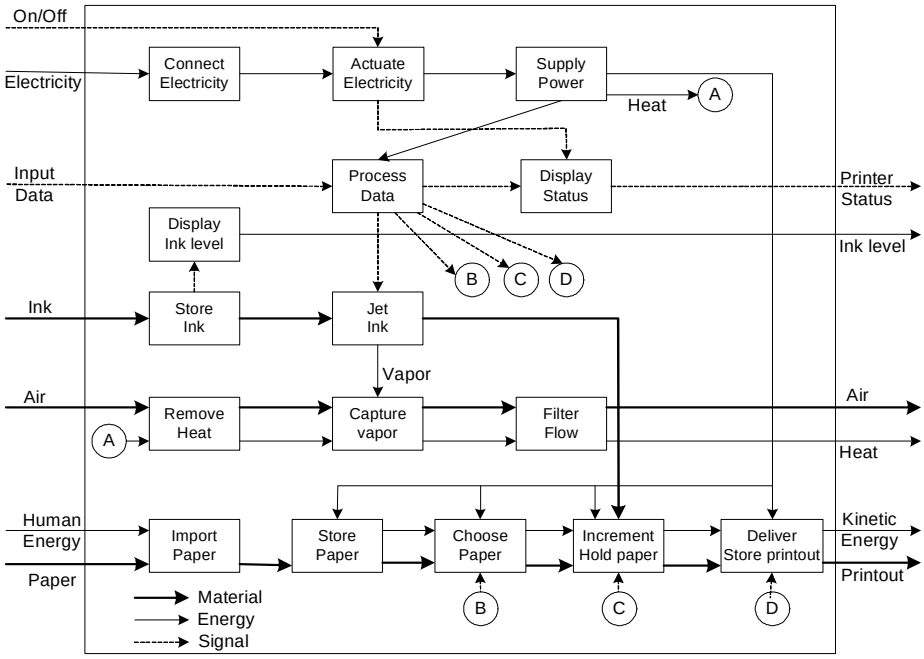


Figure 5.24: Pre-measure: (a) The original graph; (b) One configuration after one pre-measure process.

many of the same kinds of interactions with other modules, it is highly possible to build an outport inside M and make the components of M interact with other modules through the outport. The assumption in this step is that the changes happened in graph representations should not affect the functional behaviors of the original systems represented by the graph representations. For example, the electrical or signal interactions can satisfy this assumption. However, in engineering practice, for some kinds of interactions, especially those related to geometry, it is not feasible to apply this technique. In those cases, the pre-measure step should not be applied. In the HP printer example, the pre-measure is only applied to the links whose attributes are signal or electrical energy.

In Figure 5.24, module M_2 has four type t interactions with module M_1, so a new outport P is built in module M_2. Then, components A, B, C, D do not directly interact with module M_1, but through outport P. Is it always better to introduce an extra outport? In Figure 5.24 module M_2 has only two type s interactions with M_1. This raises the question of whether it is good to have an outport in this case? It is necessary to have rules to tell whether it's good or not to have an outport. One way to do this is to compare the modularity measure of the structures with or without the outport. In general, the modularity measure of all structures with different outport configurations should be calculated, and then the structure with the best modularity has the configuration of the outports.

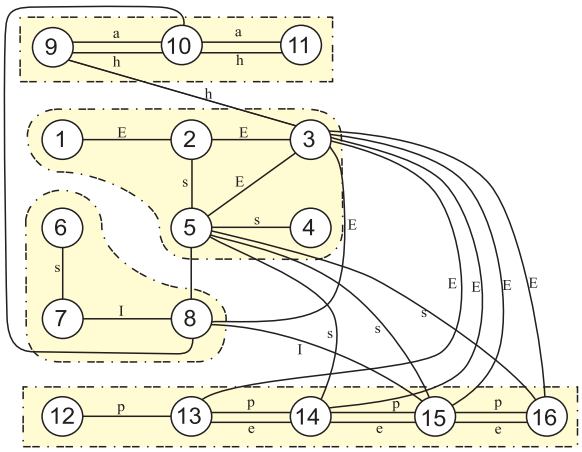


Figure 5.25: The modular decomposition of the function structure from [70].

5.5.2 Results and Discussions

In genetic computation, the population size is set to 50, the crossover rate to 0.9, the mutation rate to 0.3, and the uniformly weighted aggregation function is used. The algorithm applies a pre-measure process on every individual (genome) before it is evaluated by the MDL measure. Figure 5.23(b) shows the best modular decomposition found by the algorithm.

The result obtained by MDL-based modularity measure, shown in Figure 5.23(b), is very close to the modular decomposition [70] shown in Figure 5.25. 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 the Figure 5.25. If only looking at the abstract graphs, it is difficult to tell which decomposition is better. In real design activities, the real physical meaning and the preference of human experts affect the design results very much. How to integrate these factors into our method needs further investigation. One advantage about our technique is that it also provides some information on the configuration of the outports of modules.

5.6 Summary

Chapter 3 and Chapter 4 propose information-theoretic measures of modularity. In this chapter, the measures are used to hierarchically decompose abstract graphs and a real function structure. Some new case specific mutation and crossover operators of genetic algorithms have been developed to stochastically search hierarchical decompositions.

The work described in this chapter shows that the techniques are promising, and the information-theoretic measures are applicable for comparing hierarchical decompositions and identifying hierarchical modular structures in graph-type diagrams (structures), which are commonly used in engineering design.

The method still needs to be verified in more complex systems, and it is still worth investigating the computation complexity of the technique and scalability of the computation while the sizes of graphs or systems grow up.

Chapter 6

Modularization

6.1 Introduction

The main motivation to clarify the concept of modularity and study on modularity measures is to produce modular engineering design, especially by evolutionary design, such as truss structure synthesis [2, 42] and artificial neural network synthesis. There are many factors affecting the evolution of modular structures, such as genome representation, fitness function, learning, task structure.

If representations of genomes in an evolutionary system are designed in such a way that favors modular structures, this should definitely help to produce modular results. This kind of representation can be achieved by indirect encoding schemes which adopt a non-trivial genotype-phenotype-mapping. Most of them are grammar-based generative representations [23, 33, 45], such as L system [59, 60]. In [23, 33], the adjacency matrix of the graph describing neural networks is not directly stored in a genome, but a grammar-based generative system is used to construct the neural network from a give genotype.

Modularity-favorable fitness function can exert more selective pressure on modular structures. This can be done by integrating modularity measure into one fitness function or making modularity measure an independent fitness measure in an multi-objective genetic algorithm [16].

The structures of building blocks which final products are built greatly from affect the modularity of products. Take neural network synthesis as an example. Compared to synthesizing networks from every basic unit, connection, and node, it should be much easier to build modular networks from some higher level units which can realize some high level functions. Happel et al. [35, 36] have built Categorizing And Learning Module (CALM) from building blocks which can realize several functions, such as excitation and inhibition. Another strategy is to begin with very basic units and form modules by learning during evolutionary computations.

Another factor affecting modularity evolution is the internal structures of tasks. How does the modularity of a task affect the modularity of final design products by evolutionary computation? What kinds of tasks are modular structure-favorable?

Much work needs to be done to look into the effect of the above factors on evolving modular structures. This preliminary investigation studies the effects of modularity of tasks and tries to see if *modular problems are solved better by modular structures than non-modular structures.* The experiment is to learn two separated tasks in a two-layer feed-forward neural network.

6.2 Multi-layer Neural Networks and Backpropagation

The artificial neural networks are built from artificial neurons, which together with learning algorithms need to train such structures. An artificial neuron is an abstract model of the biological neuron. The strength of a connection is coded in the weight. The intensity of the input signal is modeled by using a real number instead of a temporal summation of spikes. The artificial neuron works in discrete time steps. The inputs are read and processed at one moment in time. The artificial neuron shown in Figure 6.1 is a very simple processing unit. The neuron has a fixed number of inputs n and an activation function f. Each input is connected to the neuron by a weighted link w_i. The neuron firstly sums up the inputs x_i according to $\sum_{i=1}^{n} x_i w_i$, then input the weighted sum into the activation function f to get the output. Two often-used kinds of activation functions are a simple threshold function and a sigmoid function. As we will see later, the backpropagation algorithm requires the function to be differentiable, so it is normal to use a sigmoid function instead of a threshold function for the activation function, which is normally wanted to be saturate at both extremes. Either a 0/1 or a ± 1 range can be used, with

$$f(h) = \frac{1}{1 + \exp(-h)}$$

and

$$f(h) = \tanh(h) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)},$$

respectively, for the activation function.

The simplest neural network is a single layer network which consisting of m neurons, each having n inputs. The type of network is widely used for linear separable problems, but a single layer network is not capable of classifying nonlinearly separable data. One very simple example of a data set which is not linearly separable is the two dimensional XOR problem. Then people introduced multi-layer networks to solve nonlinear classification problems by employing hidden layers. The additional hidden layers can be interpreted geometrically as additional hyper-planes, which enhance the separation capacity of the network. For example, a network with just one hidden layer can represent any Boolean function (including for example XOR). One typical two-layer network architecture is shown in Figure 6.2. The input vector has n dimensions, the output vector has m dimensions, the bias (the used constant input) is -1, and there is one hidden layer with l neurons. The matrix

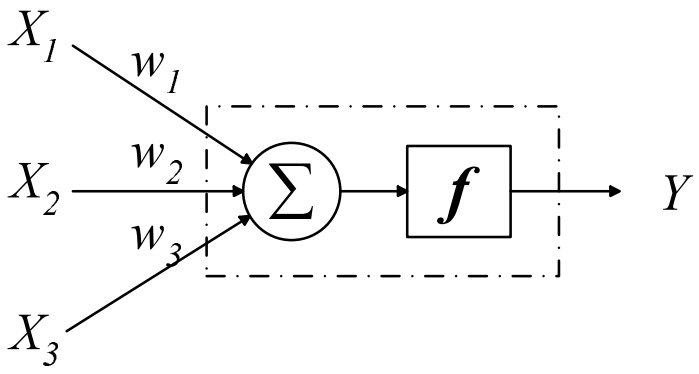


Figure 6.1: An example of artificial neuron.

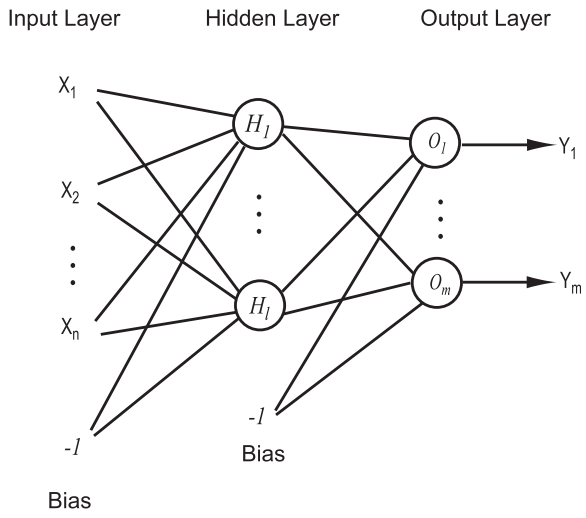


Figure 6.2: Multi-layer artificial neural network.

V holds the weights of the neurons in the hidden layer. The matrix W denotes the weights of the neurons in the output layer.

The immediate question is on the capabilities of multi-layer neural networks. It can be seen from the following *Hecht-Nielsen Theorem* that multi-layer networks are universal approximators. The *Hecht-Nielsen Theorem* states that any continuous function $f : I^n \mapsto \mathbb{R}^m$ can be approximated by a feed-forward network with n inputs, $2n + 1$ hidden neurons, and m output nodes [37, 38].

However, this theorem is only an existence theorem, and gives no assistance in how to find it. In particular, it gives no indication of a method to find the appropriate weights. Fortunately, many different learning methods have been developed. Most of the supervised methods are based on the idea of changing the weights in a direction such that the difference between the calculated output and the desired output is decreased. Examples of such rules are the Perceptron learning rule, the Hebbian learning rule, the Widrow- Hoff learning rule, and the gradient descent learning rule. The most commonly used method is the backpropagation learning algorithm, which is a gradient descent learning rule.

6.2.1 Backpropagation Algorithm

The backpropagation algorithm was invented independently by Bryson and Ho [9], Werbos [111], Parker [72], and Rumelhart, Hinton and Williams [83]. The basic idea is to present the input vector to the network; calculate in the forward direction the output of each layer and the final output of the network. For the output layer, the desired values are known, and therefore the weights can be adjusted as for a single layer network, in the case of the BP algorithm, according to the gradient decent rule. To calculate the weight changes in the hidden layer, the error in the output layer is backpropagated to these layers according to the connecting weights. This process is repeated for each sample in the training set. One cycle through the training set is called an epoch. The number of epochs needed to train the network depends on various parameters, especially on the error calculated in the output layer. The following description of the backpropagation algorithm is based on the descriptions in [75, 40]. The assumed architecture is depicted in Figure 6.2, which is a two-layer network with n inputs, l hidden neurons, and m outputs. The bias (the used constant input) is -1. For convenience, let us introduce the following notations.

- $\mathbf{x} = \{x_1, \cdots, x_n\}$ be input vector,
- $\mathbf{x}' = \{-1, x_1, \cdots, x_n\}$ be the enlarged input vector,
- $\mathbf{h} = \{h_1, \cdots, h_l\}$ be the aggregated input vector of hidden neurons,
- $\mathbf{p} = \{p_1, \cdots, p_n\}$ be the output vector of hidden neurons,
- $\mathbf{p}' = \{-1, p_1, \cdots, p_l\}$ be the enlarged aggregated output vector of hidden neurons,

- $\mathbf{q} = \{q_1, \cdots, q_m\}$ be the aggregated input vector of output neurons,
- $\mathbf{z} = \{z_1, \cdots, z_n\}$ be the output vector of output neurons
- matrix $V = [v_{ij}]_{(n+1)\times l}$ hold the weights of the hidden neurons,
- matrix $W = [w_{ij}]_{(l+1)\times m}$ hold the weights of the output neurons,
- $T = \{(\mathbf{x^1}, \mathbf{y^1}), \cdots, (\mathbf{x^d}, \mathbf{y^d})\}$ be the training set, where $\mathbf{x^i}$ is the input vector and $\mathbf{y^i}$ be the desired output vector.

Then, the backpropagation procedure is:

1. Initialize the weights V and W to small random values, and choose learning rate η.
2. Randomly choose a pair of training data $(\mathbf{x}, \mathbf{y}) \in T$.
3. Propagate the signal forwards through the network.

$$\begin{aligned} \mathbf{h} &= V^T\mathbf{x}', \qquad \mathbf{p} = g(\mathbf{h}), \\ \mathbf{q} &= W^T\mathbf{p}', \qquad \mathbf{z} = g(\mathbf{q}). \end{aligned}$$

4. Compute the deltas for the output layer and propagate the errors back-wards to compute the deltas for the hidden layers.

$$\begin{aligned} \delta_i^y &= g'(q_i)(y_i - z_i), \quad i = 1, \cdots, m, \\ \delta_i^h &= g'(h_i)\sum_{j=1}^{m} w_{ij}\delta_j^y, \quad i = 1, \cdots, l, \end{aligned}$$

   where g' is the differentiation of g.

5. Update the connection weights.

$$\begin{aligned} W &\leftarrow W + \eta\delta^y\mathbf{q}', \\ V &\leftarrow V + \eta\delta^h\mathbf{h}'. \end{aligned}$$

6. Repeat from step 2 until every pair in the training data has been used once.
7. Calculate the error by the following equation. Repeat from step 2 until error is small enough.

$$E = \frac{1}{md}\sum_{i=1}^{d}\sum_{j=1}^{m}(y_j^i - z_j^i)^2. \tag{6.1}$$

The selection of the parameters for the backpropagation algorithm and the initial settings of the weight influences the learning speed as well as the convergence of the algorithm. The initial weights

chosen determine the starting point in the error landscape, which controls whether the learning process will end up in a local minimum or the global minimum. The easiest method is to select the weights randomly from a suitable range, such as between (-1,1). If the weight values are too large, the net value will large as well; this causes the derivative of the activation function to work in the saturation region and the weight changes to be near zero. For small initial weights, the changes will also be very small, which causes the learning process to be very slow and might even prevent convergence.

The learning coefficient η determines the size of the weight changes. A small value for η will result in a very slow learning process. If η is too large, the large weight changes may cause the desired minimum to be missed. A useful range is between 0.05 and 2, dependent on the problem. An improved technique is to use an adaptive learning rate. A large initial learning coefficient should help to escape from local minima, while reducing η later should prevent the learning process from overshooting the reached minimum.

6.3 Modularity Measure

Only the topology modularity is considered in this experiment. Since in the experiments the decompositions of input nodes and output nodes are fixed, it's only necessary to consider the decompositions of hidden nodes. One possible decomposition is shown in Figure 6.3. Let L be the MDL of the structure without modular clustering and L_m be the MDL of the structure with modular architecture. Let c be a decomposition and $\mathcal{C}$ be the set of all possible decompositions. Then,

$$M = \frac{L - \min_{c \in \mathcal{C}} L_m(c)}{L}. \tag{6.2}$$

6.4 Experiments and Results

To test the hypothesis that a modular problem is solved better by modular structures, networks are trained to learn a completely separable problem. Specifically, the learning data set is generated from the following functions

$$y_1 = (x_1 \text{ and } x_2) \text{ or } x_3, \text{ and} \tag{6.3}$$

$$y_2 = (x_4 \text{ or } x_5) \text{ and } x_6. \tag{6.4}$$

It contains all possible combinations of inputs. The neural network has 6 inputs (not including the bias input), 2 outputs, and one hidden layer. The learning capability of a neural network is affected by the number of paths from inputs to outputs and the overlap among those paths,

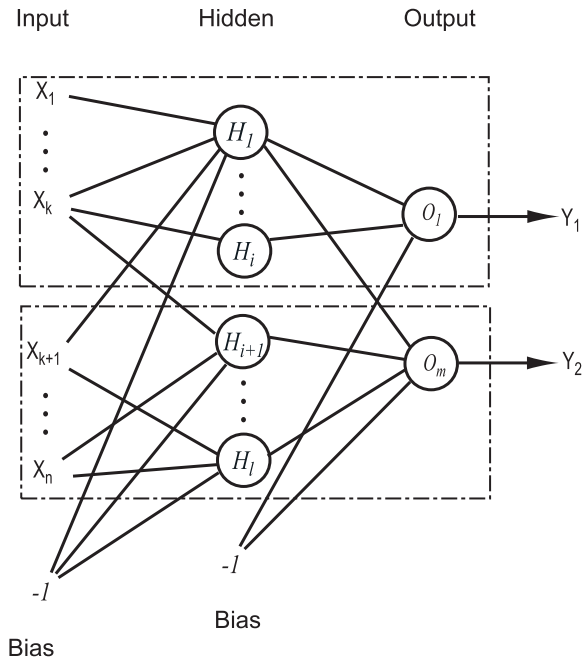


Figure 6.3: An example decomposition for modularity measure.

which is determined by the number of links and nodes, and their connectivity. The following three experiments test the hypothesis. One has a fixed number of hidden nodes; one has a fixed number of links; and the last one has a fixed number of links and nodes.

6.4.1 Experiment 1: Fixed Numbers of Links

In this case, the hidden nodes are split into three subsets: $\{1, \cdots, 10-i\}$, $\{11-i, \cdots, 10\}$, $\{11, \cdots, 20-i\}$. There are totally $20-i$ hidden nodes, and they form two coupling neural networks. Both are fully connected. One has $\{x_1, x_2, x_3\}$ as inputs, $\{1, \cdots, 10\}$ as hidden nodes, and y_1 as output. The other one has $\{x_4, x_5, x_6\}$ as inputs, $\{11-i, \cdots, 20-i\}$ as hidden nodes, and y_2 as output, as shown in Figure 6.4. The learning results are shown in Figure 6.5. The completely separable network, i.e. $i = 0$, is located at the left-most and lowest point. That is, it has the best learning errors. This means the modular problem favors modular neural networks in this case.

6.4.2 Experiment 2: Fixed Number of Hidden Nodes

This case compares the learning accuracy of a completely integral network shown in Figure 6.6(a), and a completely modular network, shown in Figure 6.6(b), which has 20 hidden neurons. As shown in Figure 6.6, the hidden neurons in the completely modular network are split into two sets. One has i nodes, and the other one has $20-i$ nodes. Since the two subnetworks learn different tasks, and it is unknown which task is more complex and therefore needs more hidden neurons, the number

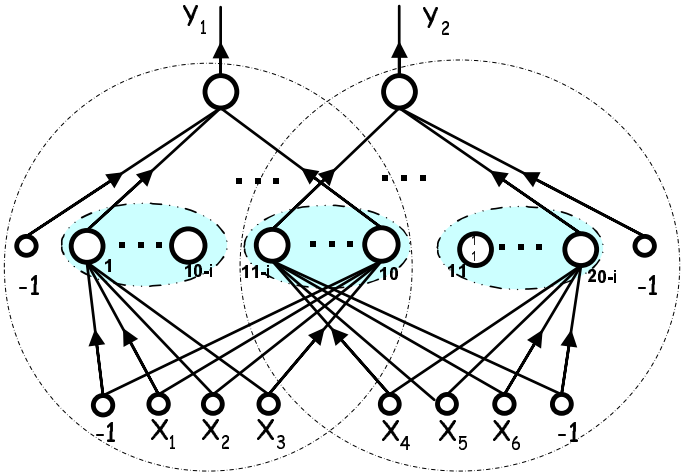


Figure 6.4: The structures of neural networks used in experiment 1.

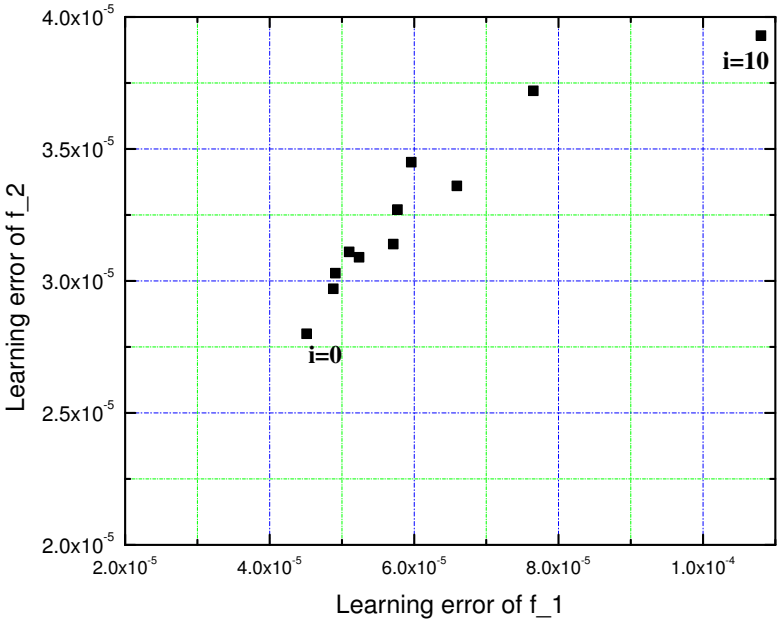


Figure 6.5: The learning errors for networks with fixed number of links.

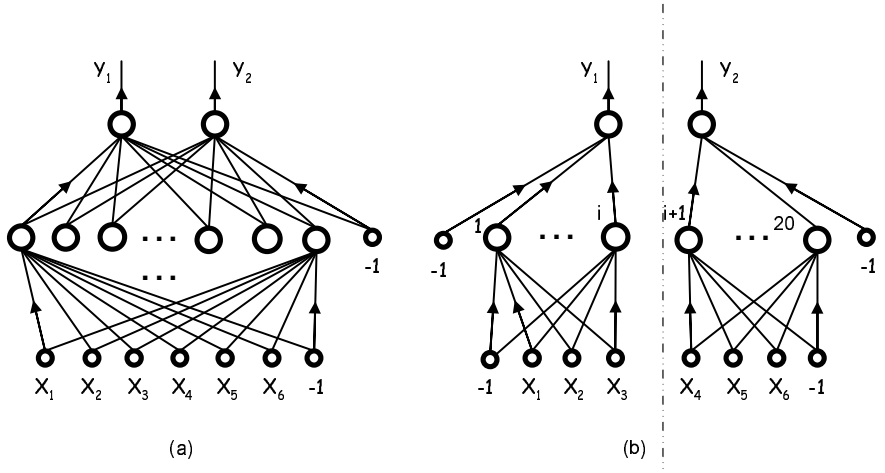


Figure 6.6: The structures of neural networks used in experiment 2.

of nodes i varies from 2 to 18. The learning results are shown Figure 6.7. It is obviously the completely integrative network gives better learning errors. Different total numbers of hidden nodes give the same results. Another result of 14 hidden nodes is shown in Figure 6.8. This preliminary result shows that a modular problem is not necessarily solved better by modular structures than non-modular structures. Then,

Do modular problems favor modularity to some extent?

This question will be investigated by evolutionary computation, which is often used to find a suitable network for a given task [112]. The experiments are performed on two-layer neural networks with 21 hidden neurons and with $tanh(x)$ as an activation function. The topology structures of networks are evolved by evolutionary computation, and the weights of connections are learned from back-propagation algorithms. The whole algorithm structure is shown in Figure 6.9.

The genome representation includes two matrices, W and V, which represent the input weights for hidden neurons and output neurons, respectively. The crossover operator is to exchange blocks in W matrices and V matrices of two parent genomes. The mutation operator is to randomly move connections (i.e., to delete a connection and insert it afterwards again at a random position). The learning error is used as the fitness function. The initial population contains 50-randomly generated individuals. A state steady genetic algorithm with gap rate 0.16 is used. In the Back-propagation algorithm, the weights are initialized with values from the interval $[-1, 1]$, and the learning rate is set to 0.1. In the computation, the modularity and number of paths from inputs to outputs are calculated for every individual in every generation.

From the results shown in Figure 6.10, the processing at the hidden layer tends to become fully distributed, and the modular problem doesn't favor modularity in this specific case.

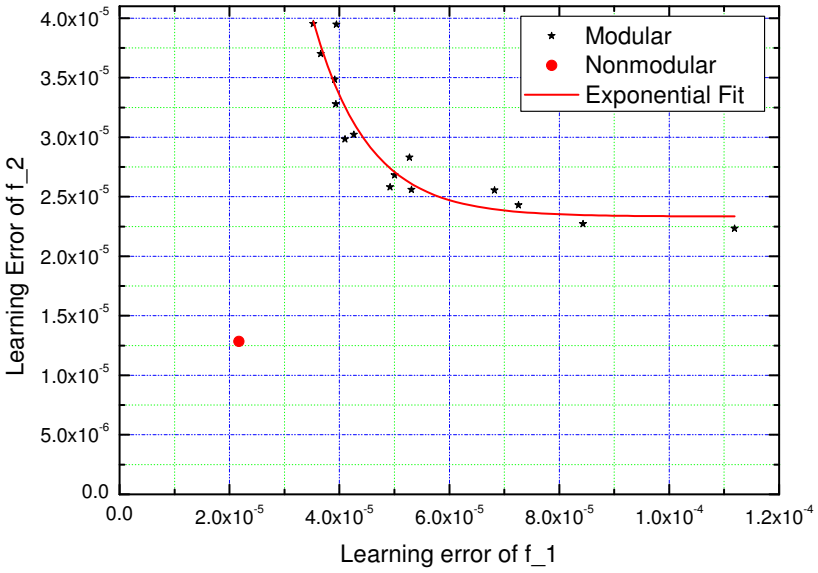


Figure 6.7: The learning errors for networks with 20 hidden nodes.

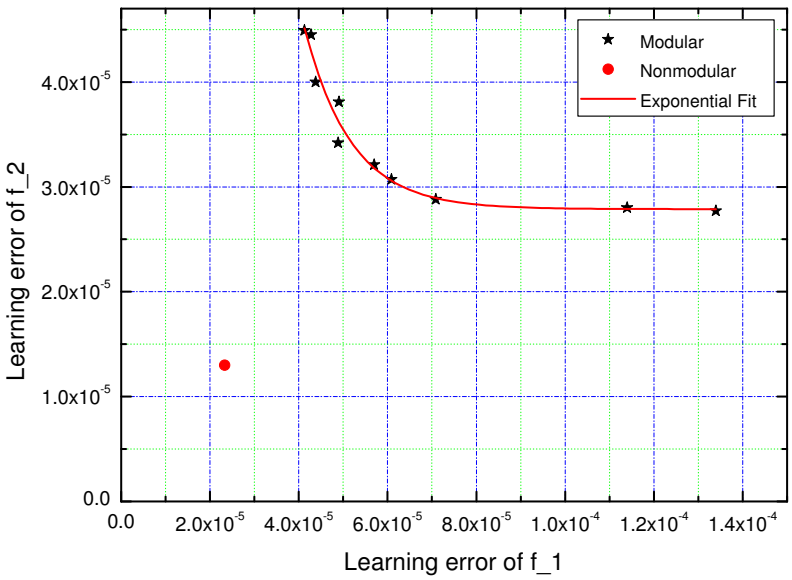


Figure 6.8: The learning errors for networks with 14 hidden nodes.

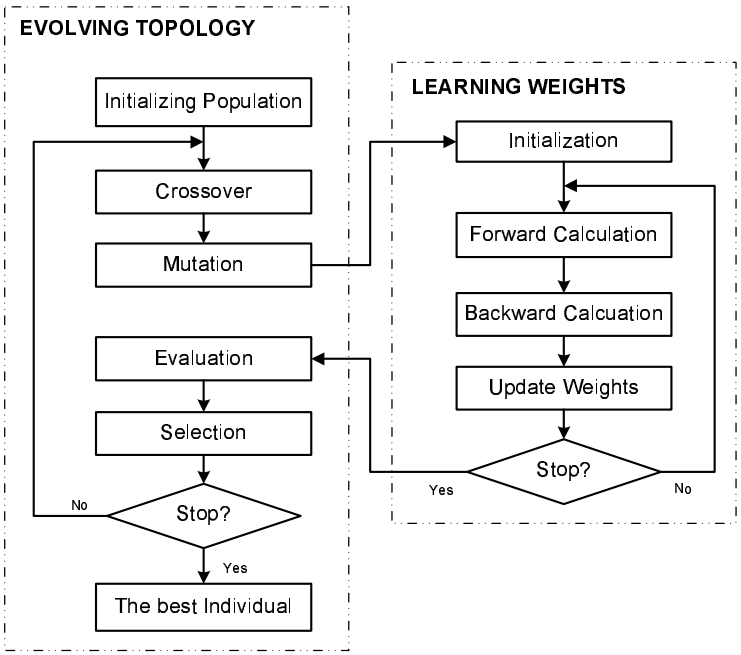


Figure 6.9: Algorithm used in experiment 2.

6.4.3 Experiment 3: Fixed Numbers of Hidden Nodes and Links

In this case, the number of hidden nodes and links are fixed to 21 and 80. Since the number of connections is fixed, the crossover operator proposed in Section 6.4.2 does not work since its two offspring have the same number of connections as the parents. Since it is not easy to design crossover operators which can guarantee the offsprings and parents have same number of connections, the evolutionary programming introduced in [22] is used here. It does not need crossover operators. The algorithm is shown in Figure 6.11. The genome representation, mutation operators, fitness function, and learning algorithm are the same as those in Section 6.4.2.

The results are shown in Figure 6.12. The increase of modularity means that modular structures are favorable in this case. While the modularity increases, the number of path almost doesn't change. This shows that the topology structures of neural networks are gradually separating during evolution, and therefore the increase of modularity.

6.5 Discussion

If neural networks are viewed as an approximator, the number of weights over the paths from inputs to outputs could be viewed as the number of coefficients of the approximator. The more the

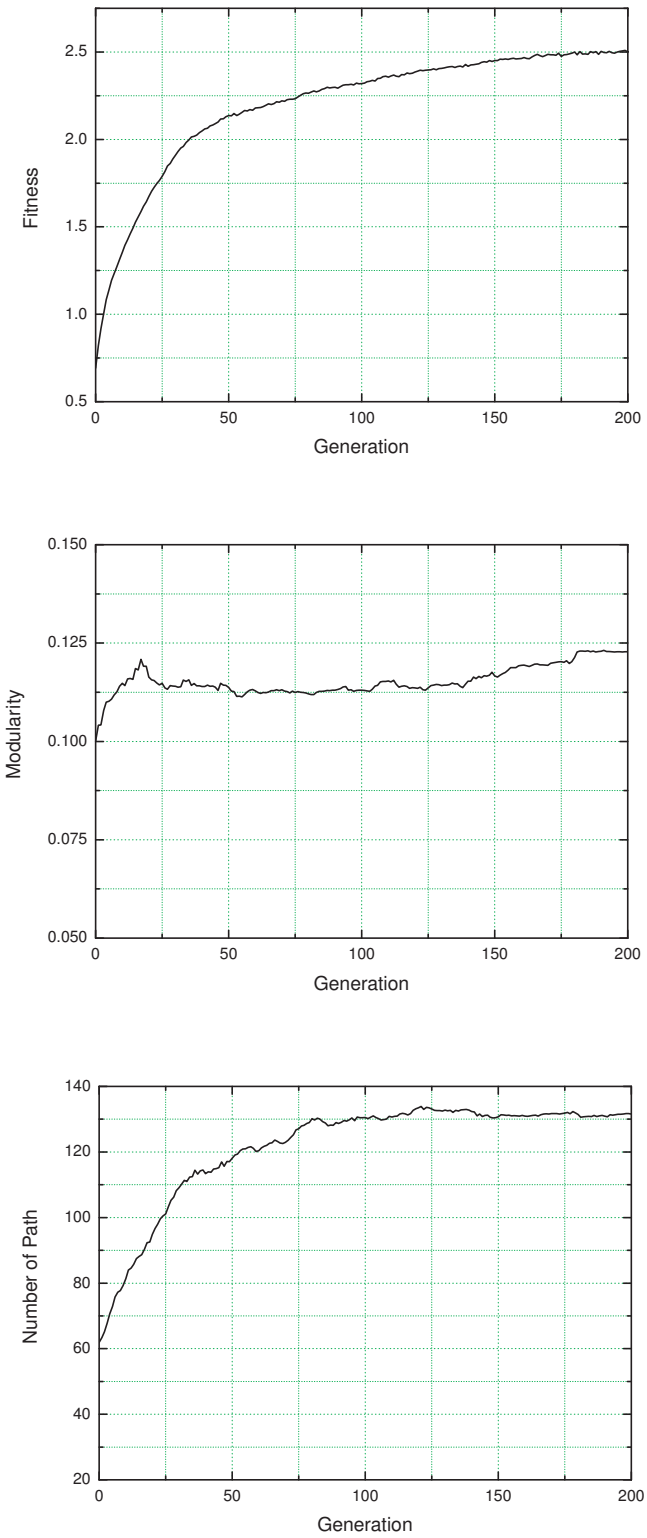


Figure 6.10: Results of evolving networks with 20 hidden nodes.

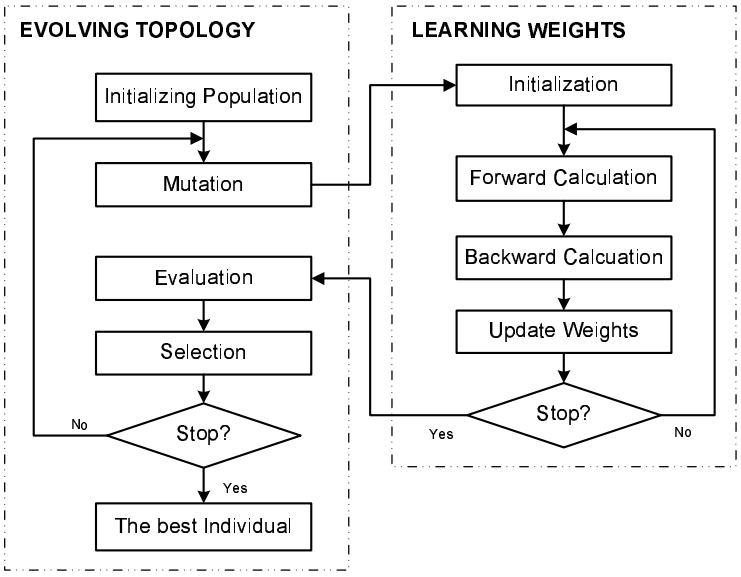


Figure 6.11: Algorithm used in experiment 3.

coefficients, the more the power of the approximator. This can explain the strong correlation between learning errors and number of paths. In Experiment 3, the number of paths has no significant change during the evolution, yet the learning error becomes better. This means that the decrease of learning errors is mainly due to the increase of modularity.

Rueckl et al. [82] studied the "what" and "where" model in similar neural networks used in Experiment 2. They showed that modular structures are favored to get better learning errors. This is different from the conclusion gotten in Section 6.4. It can be concluded that the validity of the hypothesis depends on the tasks. Usually, there are some function overlaps in two different tasks. For example, the two functions in Section 6.4 have some function overlap since they have some common data. This overlapping affects the modularity of network structures when only the learning error is used as fitness.

The validity of the hypothesis varies in the three different experiments. The links and nodes are considered as resources since the learning performance is determined by them. Then, whether a modular task is solved better by modular neural networks or non-modular ones depends on how many and what kind of resources are available for the task.

The results in Experiment 3 show that the fitness function without modularity measure can drive networks to be more modular, but the selective pressure is not very strong since the increase of modularity is slow. The possible next step is to integrate the modularity measure into fitness

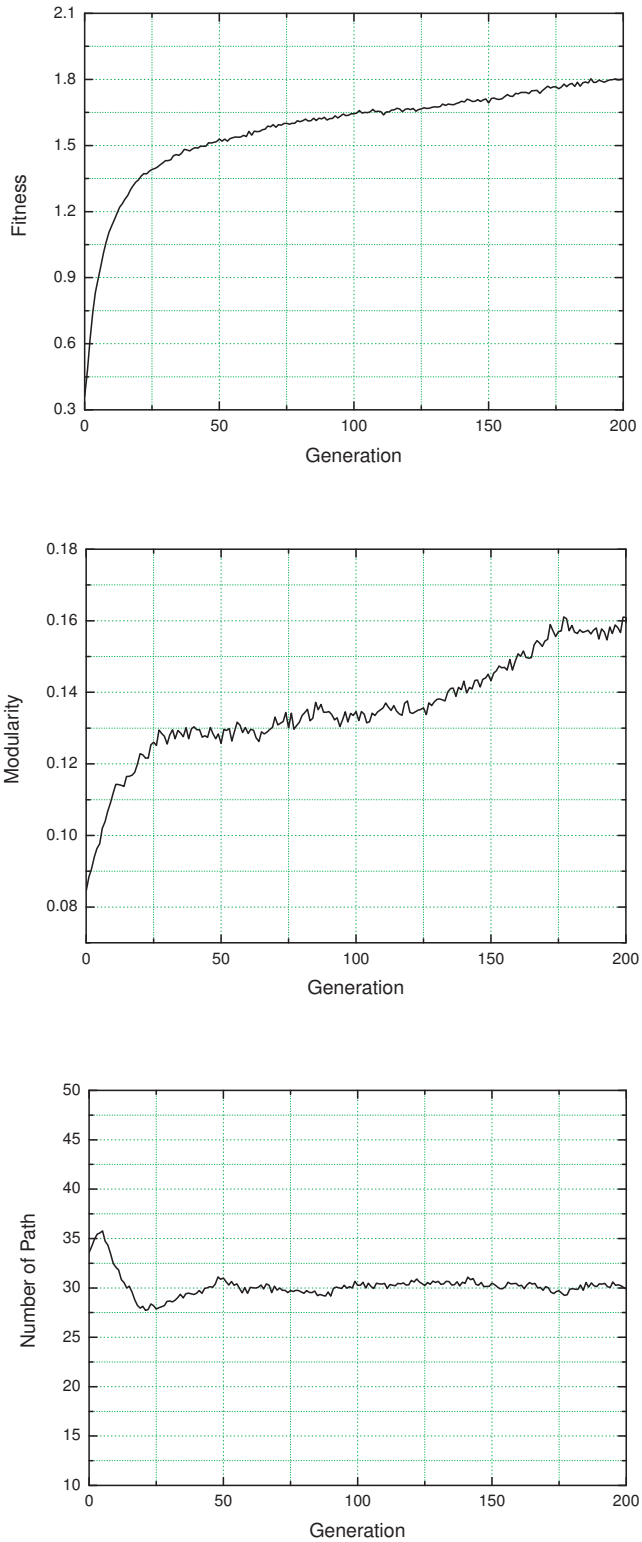


Figure 6.12: Results of evolving networks with fixed number of links.

function to increase the selective pressure.

6.6 Summary

The results in this preliminary study show that the relation between modular tasks and modular topology of feed-forward neural networks is task-dependent and depends on the amount of resources available for the tasks. The work in this chapter is a step to understand the evolution of modularity. To achieve our goal to evolve large engineering systems with modular architectures, much work can be done on the relations between the modularity evolution and factors such as genetic operators, genome representations, and fitness functions.

Chapter 7

Conclusion

7.1 Summary

Although much work has been done on modularity, most of it is qualitative and exploratory in nature. The few quantitative studies typically are function-based or design structure matrix-based, and most quantitative measures of modularity are linkage counting methods. This thesis provides a first attempt to quantify modularity from information-theoretic views and provide a more general quantitative measure of modularity.

To quantify modularity, several key characteristics of "modularity" are identified, and several questions about modularity are clarified. Specifically, they are system vs. model, specific meanings of "inter-module" and "intra-module," relativity of couplings, aggregation of modularity at different levels in a hierarchical structure, application domain of definition, and how to quantify couplings. The last question is the key to developing quantitative modularity measure.

Discussions in Chapter 2 show that modularity is actually discussed over a model of a system and, therefore, the representation of the model. According to different purposes, different aspects of design processes and products can be modeled with different representations including function structures, design structure matrices (DSM), graph representations, and systems of random variables. The graph representations can be separated into four subclasses according to whether edges of a graph are associated with attributes and whether the attributes are ordered or not. Those different representations are related. Design structure matrices can be mapped to graphs only with ordered attributes, and function structures can be mapped to graphs only with unordered attributes. By viewing adjacency matrices as covariance matrices of gaussian random vectors, a graph only with ordered attributes can be equivalent to a system of gaussian random variables.

The thesis provides two information-theoretic methods, the mutual information-based method and the Minimal Description Length (MDL)-based method to quantify modularity of these different representations. The mutual information-based method views couplings as information flow instead of real physical interactions between systems. The intuition behind this view is that the stronger

the coupling between two systems, the more information about one system can be inferred from the known information of the other system, i.e., the more information flows between them. Information flow can be quantified by mutual information, which is based on randomness and uncertainty. Since most engineering products can be modeled as stochastic systems and therefore have randomness, mutual information-based methods are applied in very general cases, and it is shown that the general existing linkage counting modularity measure is a special case of the mutual information-based modularity measure.

The mutual information-based methods are applied to products which have physical behaviors, but usually this is not the case at the early phase of engineering design, where only function diagrams exist. To exploit the benefits of modularity as early as possible in engineering design processes, an MDL-based modularity measure is proposed for structure modularity of graph structures, which can represent function diagrams. The method views modularity as a kind of regularity, which can be measured by the MDL principle.

The diagram shown in Figure 7.1 summarizes the relations of different modularity quantification methods. Systems of random variables can be quantified by the mutual information-based method; graphs without attributes, and therefore $0,1$-design structure matrices (binary design structure matrices), can be quantified by both methods and the demonstration in Chapter 5 shows that the they are equivalent in this case; graphs only with ordered attributes, and therefore function structures, can be quantified by the MDL-based method; and graphs only with ordered attributes, and therefore design structure matrices, can be quantified by the mutual information-based method. Unfortunately, both methods do not work well for graphs with both unordered and ordered attributes. To get design structure matrices, it is required to quantify the interactions between different components or design parameters. Generally the quantifying is done by designers' subjective knowledge and experience. The mutual information-based method can provide a formal and quantitative way to get design structure matrices.

Both methods have been verified to hierarchically decompose abstract graph structures and a real function structure of an HP printer. Due to the complexity of graph decomposition, genetic algorithms are used, and new genetic operators for tree representation are developed to find the optimal hierarchical decomposition.

The main motivation to quantify modularity is to produce modular engineering designs, especially by evolutionary design. There are many factors affecting the evolution of modular structures, such as genome representation, fitness function, learning, and task structure. The preliminary work in this thesis looks into how the modularity of tasks affect the modularity of products produced by evolutionary computation. Using neural networks as examples, the study shows that the relation between modular tasks and modular topology of feed-forward neural networks is task-dependent and depends on the amount of resources available for the tasks.

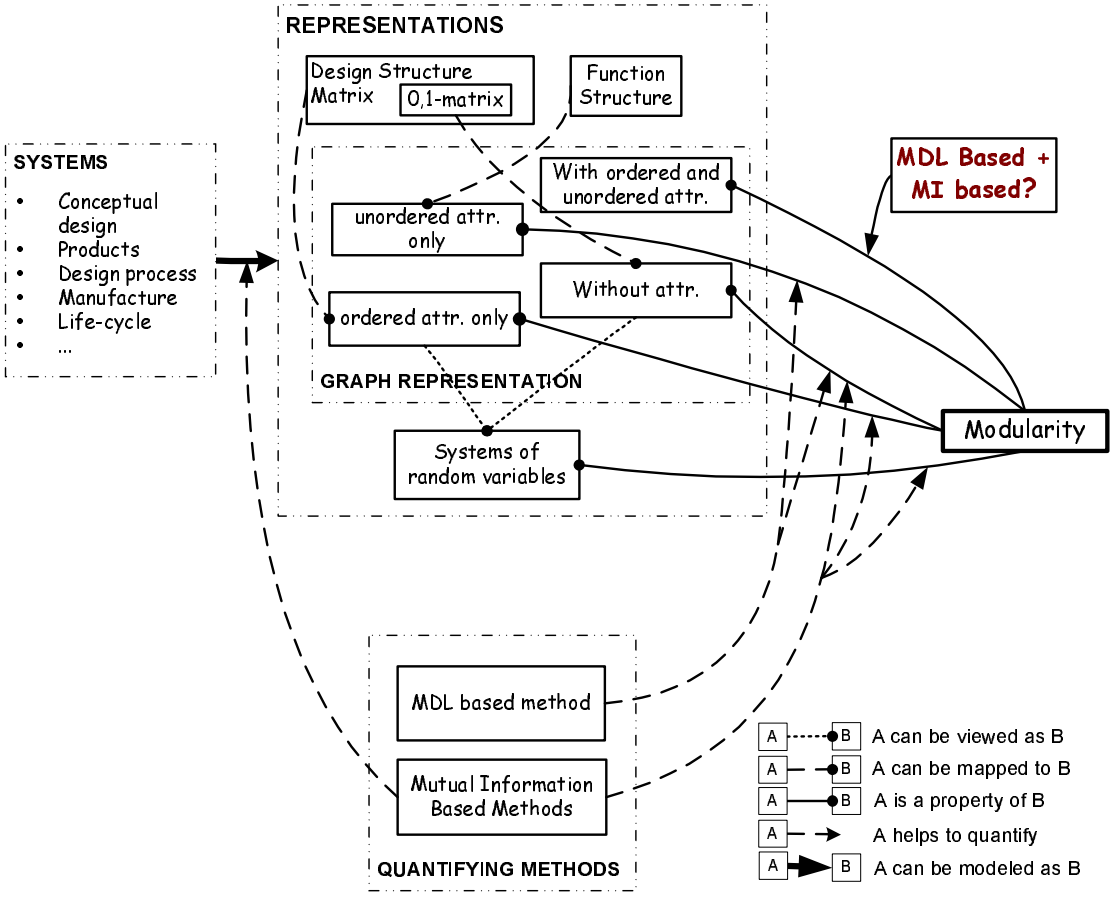


Figure 7.1: Summary of the modularity measure methods.

7.2 Future Work

One theoretical work is to remove the question mark in Figure 7.1, that is, to find a quantitative measure of the modularity of graphs with both unordered and ordered attributes. One possible approach is to integrate mutual information-based measure and MDL-based measure. It is known that a mutual information-based measure can quantify the modularity of graphs only with ordered attributes, and the MDL-based method works well for unordered attributes. The whole set of information of graphs with both unordered and ordered attributes can be separated into two sets. One only includes ordered attributes, which can be quantified by the mutual information-based method, and the other one only has unordered attributes, which can be measured by MDL-based method. The integration of the two measures, and their separate information, needs to be investigated.

The framework of mutual information-based methods developed in Chapter 3 is based on systems of random variables. How can real physical systems, such as design products, design processes, and manufacturing processes, be modeled as systems of random variables? Chapter 3 has provided some examples to explain this. It is still necessary to extend those methods to more complex systems. Sometimes it is quite difficult to get probability distributions for random variables and sometimes systems are very complex, so it is useful to provide some approximate methods to mutual information-based methods though they are theoretically natural and beautiful. An approximate method could approximate probability distributions, or mutual information, or even mutual information-based methods. One possible approximation is to use $0, 1$-function and consider the number of interactions. If there is an interaction between two components, the strength of interaction is 1, and otherwise the strength is 0. This method is oversimplified. Are there any intermediate approximations, and if so, what are they?

Another problem is relates to aggregation. As pointed out in Chapter 2, since modularity is hierarchical, the modularity of a system is an aggregation of the modularity at different levels of the system. There are many different ways to aggregate information. What is the best aggregation? What could be the criteria? How is the modularity measure affected by aggregation functions? One possible way to avoid aggregation is to use a sequential method: to decompose systems by several sequential steps and one level at one step. How is the sequential method different from the one step but multi-level method?

In computational aspects, due to complexity of clustering, evolutionary computations can be used. There are no existing genetic operators available for the special tree representations of modular structures. Though some operators have been designed in Chapter 5, it is still useful to design better operators to more efficiently and quickly explore the space of all possible decompositions. It is worth investigating the computation complexity of the techniques and scalability of the computation while the sizes of systems grow up.

The last but largest problem is to investigate how to produce modular structures in engineering design, especially in evolutionary design. It is worth applying biological understandings of modularization to engineering modularization. On the other hand, engineering study could also provide some thoughts on biological modularity since the biological modularization is still far from being fully understood. It could be useful to look into how genome representation, fitness function, learning and task structures affect modularization in evolutionary design, how to integrate modularity measure into fitness measures, and how to develop generative genome representations.

Bibliography

[1] C. Alexander. *Notes on the Synthesis of Form.* Harvard University Press, Cambridge, MA, 1964.

[2] E. K. Antonsson, F. Nicaise, and T. Honda. Configuration synthesis of a variable complexity truss. In *15th International Conference on Engineering Design.* The Design Society, Aug. 2005.

[3] C. Y. Baldwin and K. B. Clark. *Design Rules, Volume 1, The Power of Modularity.* MIT Press, Cambridge MA., 2000.

[4] C. Y. Baldwin and K. B. Clark. Managing in the age of modularity. *Harvard Business Review*, pages 84–93, Sept.-Oct. 97.

[5] D. H. Ballard. Modular learning in hierarchical neural networks. In E. L. Schwartz, editor, *Computational Neuroscience.* MIT Press, 1990.

[6] O. E. Barndorff-Nielsen and D. R. Cox. *Inference and Asymptotics.* Chapman and Hall: London, 1989.

[7] F. H. Bennett III and E. G. Rieffel. Design of decentralized controllers for self-reconfigurable modular robots using genetic programming. In *The Second NASA/DoD Workshop on Evolvable Hardware*, pages 43 – 52, July 2000.

[8] J. A. Bolker. Modularity in development and why it matters to evo-devo. *Amer. Zool.*, 40:770–776, 2000.

[9] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* New York: Blaisdell, 1969.

[10] J. Bryson. *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents.* Ph.D. thesis, MIT, 2001.

[11] J. Bryson and L. A. Stein. Modularity and specialized learning: Mapping between agent architectures and brain organization. *Lecture Notes in Computer Science*, 2036:98–??, 2001.

[12] R. Calabretta, S. Nolfi, D. Parisi, and G. Wagner. Duplication of modules facilitates the evolution of functional specialization. *Artificial Life*, 6:69–84, 2000.

[13] W. Callebaut and D. Rasskin-Gutman. *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. The MIT Press, 2005.

[14] K. B. Clark. The interaction of design hierarchies and market concepts in technological evolution. *Research Policy*, 14:235–251, 1985.

[15] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[16] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley, 2001.

[17] R. Descartes. *A Discourse on Method: Meditations on first Philosophy*. tranlated by Donal A. Cress, Hackett Pub. Co., Indianapolis, 1993.

[18] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness - A Connectionist Perspective on Development*. MIT Press, Cambridge, Massachusetts, 1996.

[19] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. Gebala. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1):1–13, 1994.

[20] S. J. Fenves and F. H. Branin. Network topological formulation of structural analysis. *ASCE, Journal of The Structural Division*, 89(4):483–514, 1963.

[21] J. Fodor. *The Modularity of Mind*. Cambridge MA: MIT Press, 1983.

[22] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, 1995.

[23] C. M. Friedrich and C. Moraga. An evolutionary method to find good building-blocks for architectures of artificial neural networks. In *Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU'96*, pages 951–956, Granada, Spain, 1996.

[24] G. D. Galsworth. *Smart, Simple Design*. Essex Junction, Vermont, Oliver Wright Publications, 1994.

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

[26] J. L. Garfield. *Modularity in Knowledge Representation and Natural-Language Understanding*. MIT Press, Cambridge, Massachusetts, 1987.

[27] J. K. Gershenson, G. J. Prasad, and S. Allamneni. Modular product design: A life-cycle view. *Journal of Integrated Design and Process Science*, 3(4), 1999.

[28] J. K. Gershenson, G. J. Prasad, and Y. Zhang. Product modualrity: Definitions and benefits. *Journal of Engineering Design*, 14(3):295–313, 2003.

[29] J. K. Gershenson, G. J. Prasad, and Y. Zhang. Product modularity: Measures and methods. *Journal of Engineering Design*, 15(1):33–51, 2004.

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

[31] M. A. Goodale and A. D. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15:20–25, 1992.

[32] F. Gruau. Genetic synthesis of modular neural networks. In *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 318–325. Morgan Kaufmann, 1993.

[33] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–183, 1995.

[34] F. Guo and J. K. Gershenson. A comparison of modular product design methods based on improvement and iteration. In *16th International Conference on Design Theory and Methodology (DTM)*. ASME, Sept. 2004.

[35] B. L. M. Happel and J. M. J. Murre. Designing modular neural network architectures using a genetic algorithm. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks*, volume 2, pages 1215–1218. Elsevier Science Publishers, 1992.

[36] B. L. M. Happel and J. M. J. Murre. Design and evolution of modular neural network architectures. *Neural Networks*, 7:985–1004, 1994.

[37] R. Hecht-Nielsen. Kolmogorovo's mapping neural networks existence theorem. In *Proceedings of the First IEEE International Conference on Neural Networks*, volume 3, pages 112–114, San Diego, 1987.

[38] R. Hecht-Nielsen. Theory of backpropagation neural networks. In *Proceedings of the First IEEE International Conference on Neural Networks*, volume 1, pages 593–605, Washington, 1989.

[39] R. M. Henderson and K. B. Clark. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35:9–30, 1990.

[40] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to The Theory of Neural Computation*. Perseus Books Publishing, 1991.

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

[42] T. Honda, F. Nicaise, and E. K. Antonsson. Synthesis of structural symmetry driven by cost saving. In *31st Design Automation Conference (DAC)*. ASME, Sept. 2005.

[43] R. Hooper and D. Tesar. Computer-aided configuration of modular robotic systems. *Computing & Control Engineering Journal*, 5(3):137–142, June 1994.

[44] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[45] G. Hornby, H. Lipson, and J. Pollack. Evolution of generative design systems for modular physical robots. In *International Conference on Robotics and Automation*, 2001.

[46] J. Hsuan. Impacts of supplier-buyer relationships on modularization in new product development. *Eur. J. Purchas. Supply Manage.*, 5(3):197–209, 1999.

[47] http://invention.psychology.msstate.edu.

[48] C. C. Huang and A. Kusiak. Modularity in design of products and systems. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 28(1):66–77, JANUARY 1998.

[49] A. Hyvärien, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley Interscience, 2001.

[50] J. W. Kalat. *Biological Psychology*. Brooks/Cole Publishing Company, 1992.

[51] A. Karmiloff-Smith, J. Grant, and I. Berthoud. Language and williams syndrome: how intact is 'intact'? *Child Developement*, 68:246–262, 1997.

[52] D. Karnopp and R. Rosenberg. *System Dynamics: a Unified Approach*. John wiley & sons, 1975.

[53] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, Oxford, 1993.

[54] S. A. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128:11–45, 1987.

[55] A. Kaveh. Graphs and structures. *Computers and Structures*, 40:893–901, 1991.

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

[57] R. N. Langlois. Modularity in technology, organization, and society. In *Management Section session on modularity, INFORMS Philadelphia Meeting*, Philadelphia, November 8 1999.

[58] R. N. Langlois and P. Robertson. Networks and innovation in a modular system: Lessons from microcomputer and stereo components industries. *Research Policy*, 21(4), 1992.

[59] A. Lindenmayer. Machematical models for cellular interactions in development, part I, filaments with one-sided inputs. *J. Theor. Biol.*, 18:280–299, 1968.

[60] A. Lindenmayer. Machematical models for cellular interactions in development, part II, simpe and branching filaments with two-sided inputs. *J. Theor. Biol.*, 18:300–315, 1968.

[61] M. Lundqvist, N. Sundgren, and L. Trygg. Remodularization of a product line: Adding complexity to project management. *J. Prod. Innov. Manage.*, 13(3):311–324, 1996.

[62] R. J. McEliece. *The Theory of Information and Coding, 2nd ed.* Cambridge: Cambridge U. Press, 2002.

[63] J. H. Mikkola and O. Gassmann. Managing modularity of product architectures: Toward an integrated theory. *IEEE Transactions on Engineering Management*, 30(2):204–218, 2003.

[64] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[65] K. S. Miller. *Multidimensional Gaussian Distributions*. New York: John Wiley, 1964.

[66] M. Mishkin, L. G. Ungerleider, and K. A. Macko. Object vision and spatial vision: Two cortical pathways. *Trends in Neurosciences*, 6:414–417, 1983.

[67] P. J. Newcomb, B. Bras, and D. W. Rosen. Implications of modularity on product design for the life cycle. *Journal of Mechanical Design*, 120(3):483–491, 1998.

[68] K. N. Otto. *A Formal Represnentational Theory for Engineeirng Design*. Ph.D. thesis, California Institute of Technology, Pasadena, CA, June 1992. 1992.

[69] K. N. Otto and E. K. Antonsson. Trade-off strategies in the solution of imprecise design problems. In T. Terano et al., editor, *In Fuzzy Engineering toward Human Friendly Systems: Proceedings of the International Fuzzy Engineering Symposium 91*, volume 1, pages 422–433, 1991.

[70] K. N. Otto and K. Wood. *Product Design*. Prentice Hall, 2001.

[71] G. Pahl and W. Beitz. *Engineering Design: A Systematic Approach*. Springer, 1996.

[72] D. B. Parker. Learning logic. Technical Report Technical Report TR-47, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

[73] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 5:1053–1058, 1972.

[74] D. L. Parnas. A technique for software module specification with examples. *Communications of the ACM*, 15:330–336, 1972.

[75] D. W. Patterson. *Artificial Neural Networks. Theory and Application*. Prentice Hall, Singapore, 1996.

[76] E. Pöppel. Temporal mechanisms in perception. *International Review of Neurobiology*, 37:185–202, 1994.

[77] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[78] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Ann. Statist.*, 11:416–431, 1983.

[79] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[80] J. Rissanen. Hypothesis selection and testing by the MDL principle. *Computer Journal*, 42:260–269, 1999.

[81] D. Robertson and K. Ulrich. Planning for product platforms. *Sloan Manage. Rev.*, pages 19–31, 1998.

[82] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are "what" and "where" processed by separate cortical visual systems? a computational investigation. *Journal of Cognitive Neuroscience*, 1:171–186, 1989.

[83] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Nature*, 323:533–536, 1986.

[84] R. Sanchez. Modular architectures in the marketing process. *Journal of Marketing*, 63(Special Issue):92–111, 1999.

[85] R. Sanchez and J. T. Mahoney. Modularity, flexibility, and knowledge management in product and organisation design. *Strategic Management Journal*, 17 (Winter Special Issue):63–76, 1996.

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

[87] M. J. Scott and E. K. Antonsson. Aggregation functions for engineering design trade-offs. *Fuzzy Sets and Systems*, 99(3):253–264, 1998.

[88] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[89] D. M. Sharman, A. A. Yassine, and P. Carlile. Architectural optimisation using real options theory and dependency structure matrices. In *Proceedings of the ASME DETC DAC 2002*, 2002.

[90] H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[91] S. Sosale, M. Hashemian, and P. Gu. An integrated modular design methodology for life-cycle engineering. *Annals of the CIRP*, 46, January 1997.

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

[93] D. V. Steward. The design structure system: A method for managing the design of complex system. *IEEE Transactions in Engineering Management*, 28(3):71–84, 1981.

[94] R. Stone, K. Wood, and R. Crawford. A heuristic method to identify modules from a functional description of a product. In *Proceedings of DETC98*, Atlanta, GA, 1998.

[95] W. A. Swan. Proposes standardization of car sizes. *The Automobile*, 31, 1914.

[96] P. Testa, U. O'Reilly, M. Kangas, and A. Kilian. Moss: Morphogenetic surface structure: A software tool for design exploration. In *Proceedings of Greenwich 2000: Digital Creativity Symposium*, pages 71–80, London, UK, January 2000.

[97] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, 2nd Edition*. Wiley Interscience, 2003.

[98] K. T. Ulrich. The role of product architecture in the manufacturing firm. *Research Policy*, 24:419–440, 1995.

[99] K. T. Ulrich and W. P. Seering. Function sharing in mechanical design. *Design Studies*, 11:223–234, 1990.

[100] K. T. Ulrich and K. Tung. Fundamentals of product modularity. In *ASME Issues in Design/Manufacture integration*, pages 73–80, Atlanta, USA, 1991.

[101] G. P. Wagner. Homologues, natural kinds and the evolution of modularity. *American Zoologist*, 36:36–43, 1996.

[102] G. P. Wagner, J. Mesey, and R. Callabretta. Natural selection and the origin of modules. In W. Callebaut and D. Raskin-Gutman, editors, *Modularity: Understanding the Development and Evolution of Complex Systems*. MIT Press, Cambridge, MA, 2002.

[103] G.P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50:967–976, 1996.

[104] C. S. Wallace and D. M. Boulton. An information theoretic measure for classification. *Computer Journal*, 11:185–194, 1968.

[105] C. S. Wallace and D. L. Dowe. Minimum message length and kolmogorov complexity. *Computer Journal*, 42:270–283, 1999.

[106] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding (with discuss). *J.R.Statist Soc. B*, 49:240–265, 1987.

[107] B. Wang and E. K. Antonsson. Information measure for modularity in engineering design. In *16th International Conference on Design Theory and Methodology (DTM)*. ASME, Sept. 2004.

[108] B. Wang and E. K. Antonsson. Hierarchical modularity: Decomposition of function structures with minimal description length principle. In *17th International Conference on Design Theory and Methodology (DTM)*. ASME, Sept. 2005.

[109] R. A. Watson. Hierarchical module discovery. In H. Lipson, E.K. Antonsson, and J.R. Koza, editors, *AAAI Spring Symposium - Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 262–267, 2003.

[110] R. A. Watson. Modular interdependency in complex dynamical systems. In Dominique Gross, Tom Lenaerts, and Richard Watson, editors, *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*, pages –, UNSW Australia, December 2003.

[111] P. J. Werbos. *Beyond Regression: New Tools for Predication and Analysis in the Behavioral Sciences*. Ph.D. thesis, Harvard University, 1974.

[112] X. Yao. Evloving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[113] T. L. Yu, A. Yassine, and D. E. Goldberg. A genetic algorithm for developing modular product architectures. In *Proceedings of the ASME DETC DTM 2003*, 2003.