IJAMS International Journal of Advanced Manufacturing Systems

Product Module Partition and Optimization for Customization

Yiliu Liu, Y. -S. Ma*, Stephen S.G. Lee and Roger J. -X. Jiao
School of MAE, Nanyang Technological University, Singapore 639798

Abstract: Modularity design helps mass customization to satisfy customers' requirements in a reasonable cost level with the advantages of interchangeable modules, increased variety, and reusability. Usually, after classifying product design space into three domains, the functional, the technological and the physical models, and identifying mappings among them, numerous partitioned modules are generated in a product, such as basic modules, sub-modules and advanced modules, and their different combinations. This paper proposes a mathematic method to seek the optimum partition considering the needs of customers and the aforementioned modularity requirements. A genetic algorithm is applied. Finally, a case about personal computers is studied to indicate effect of the algorithm.

Keywords: Product modularity; Mass customization; Modularity optimization; Product lifecycle management

1. Introduction

The manufacturing industry is undergoing a major shift from traditional mass production to mass customization (Pine, 1993; Silveira et al, 2001). The goal of this new production mode is to provide personalized products with relatively low cost. According to Gaimon and Singhal (1992), the design stage has the most influence on the whole lifecycle. Modular design can resolve the competing goals of low cost, high variety and flexibility in customization. Modular design emphasizes using interchangeable and configurable modules or components with 'mix-and-match capability' (Balwin and Clark, 1997). These modules or components are distinct, standardized building blocks that can be produced independently of one another (Fixson, 2002; Mikkola, 2001; Kusiak and Huang, 1996, 1998) but, assembled together into a bundle of features that characterize the product.

Some potential benefits of modular design include ease of change of design, shortened order lead-time, decoupled risks, effective and efficient product diagnosis, maintenance, repair and disposal (Kusiak and Huang, 1996), ease of assembly and disassembly, and module reuse (Scheidt ant Zong, 1994). Because optional product functions may be realized by re-configuring the modules, the customer has a greater variety of product choices (Sosale at al, 1997).

While much research about modularity has been carried out (Ulrich, 1995), how to partition a product on modularity for customization is still a question to answer. Most reported partition methods lack enough considerations about customization (Kamrani and Gonzalez, 2003; Kreng and Lee, 2004). This paper focuses on modularity partition hierarchical structure, and optimization according to customers' requirements. A new optimization method is introduced. The paper is organized as follows: Section 2 introduces the product domains and mappings. Section 3 presents the module partition. We provide objective functions and their optimal process in Section 4. Section 5 studies a case, and the conclusions are presented in Section 6.

2. Product Modules and Mappings

Modularity design aims to meet the need of customers, so designers have to combine many components and modules to achieve the desired functions. Based on customers' requirement, most product descriptions can be classified into three models (Albano and Suh, 1992), the functional, the technological and the physical ones. These models act as a backbone of product information (Erens and Verhulst, 1997). The functional model addresses the customer requirement in the form of declarative functional specifications; the technological one addresses the design objectives through the interactions among components or modules, i.e. conceptual design features; the physical one materializes the practical solution by creating the detailed design model, with a hierarchical structure of modules, components and detailed design features (Chen et al. 2006).

This work assumes that a module is a collection of components, either one or more, that demonstrates the maximum commonalty and similarity in product structure according to the constraints of product variety, cost, customer requirement, supply chain, reusability, end-of-life, etc (Jiao et al., 2006). To study the mapping relations between functional and physical models, Ulrich (1995) proposed 4 types of mapping, i.e. one-to-one, one-to-many,

* Corresponding author. Address: School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. Tel: (+65) 67905913. Fax: (+65) 67911859. email: mysma@ntu.edu.sg

many-to-one, and many-to-many types. However, technological model was not mentioned. Clearly, between the functional and technological models, for each function, there must be at least one, or sometimes the combination of several mechanisms, is needed to meet customers' requirement. Hence, a *1:N* mapping from a function to its corresponding mechanisms exists, in which $1 \leq N \leq n$ (*n* is the total number of all mechanisms used in the product). On the other hand, it is also possible to have one mechanism to address multiple functions. Therefore complex many-to-many (*X:Y* as shown in Figure 1) relations exist. Here, *Y* technological features may be needed to satisfy *X* functions. Similarly, *Z* modules may be needed in the physical model, and a *Z:X* mapping can be created to the functions according to customer requirements. Theoretically, there is no unambiguous mathematical relationship on the desired level of modularity (Fig. 1).

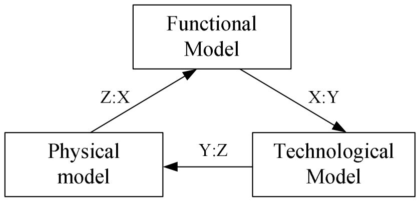

Figure. 1 Mappings between product models

3. Modules Partition

As discussed in Section 2, for *X* functions required by customers, *Z* modules are required in the product to carry out. When the requirement of customers, *X*, remains constant, the change of number of modules, *Z*, in a product, leads to the change of the ratio of *Z:X*. Hence, this provides the optimization space for customers' satisfying degree in some extent. Based on different module partition methods, an optimized ratio between physical model and functional one should be found, and this section describes the pre-requisite and the method for module partition.

Traditionally, modularity is viewed as depending on two characteristics: similarity between the physical and functional architecture of the design; and minimization of incidental interactions between physical components (Ulrich, 1995). Using typical a mechatronic product as an example, Kusiak and Huang (1996) highlighted two types of relationships involved in the modularity concept: high degree of functional interactions of inter-modules; and low degree of interactions among components of intra-modules. Here this paper introduces a concept called *basic module*. A basic module possesses the following attributes:

Substantiality: Basic modules have certain volume, shape, weight; have cost whether producing alone or purchasing; consume energy when using; will be wear to failure unavoidably;

Independence: Basic modules can exist to deliver the required functions without depending upon other parts of the product. Naturally, they can be purchased separately and can be detached without influencing others;

Functionality: Basic modules demonstrate the functional characteristics according to customers' requirements.

Relevance: Different modules take input from and provide output to others.

In addition, commonly a product consists of a hierarchical structure of modules with different layers and granularity. Therefore, it is necessary to define the relations among modules. First, two module concepts are introduced: *sub-module* and *advanced module*. In many cases, some modules in a product may exist as an inter-exchangeable product, and the motherboard of a Personal Computer (PC) is an excellent example, because it is often taken as a module of the whole PC. However, it possesses several sub-modules as well, such as BIOS chip set, the CPU socket, I/O units, the USB/IEEE1394 controlling chip, memory slots, the AGP slot, PCI slots, the power circuit, hardware monitor, etc.. These *sub-modules* can be regarded as the 2nd-level modules of the whole product. Sub-modules in a product hence have a nesting hierarchical structure. In another case, under the functional and physical constraints among these sub-modules, they can be combined together to form larger modules. For instance, the motherboard and network card are sub-modules of a computer, but they will become one module when manufacturers combine them together for production and sales. To distinguish from basic modules, we name this kind of modules as *advanced modules*. Clearly, these three kinds of modules are the same in essence, and the difference among them is the relative levels they lie in the product. When several modules are grouped into a new module, the new one is an advanced module in relation to its members; and if we view the new one as a basic module, its members become sub-modules.

Every two basic modules can be combined according to the following two principles: one module's functionality requires the other; these two modules are physically interconnected. Suppose there are *n* basic modules in a product, and then the connection matrix of the *n* modules is as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1j} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2j} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} & \cdots & r_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nj} & \cdots & r_{nn} \end{bmatrix}, \qquad (1)$$

where r_{ij} represents the connection between *i*th module and j^{th} module, $r_{ij}=0$ or 1. That $r_{ij}=0$ means there is no connection between i^{th} module and j^{th} module, on contrast, $r_{ij}=1$ means there is a connection, and they can be grouped into an advanced module. *R* is a symmetric matrix, namely, $r_{ij}=r_{ji}$. Note that because this matrix just shows the connections among modules in the final product, but it does not indicate that those connected modules will necessarily be combined into a new advanced module. It should be distinguished from the combination matrix that is to be introduced in Section 4.2. This paper focuses on how to combine suitable number of basic modules into advanced modules. In general, the cost of producing advanced modules should be less than sum of that of producing two modules respectively. However, such module partition involves more aspects of considerations, such as interface constraints among modules, product varieties, upgrading abilities, purchasing bundling, etc.; it also influences the satisfying level of customers.

4. Objective Function and Optimization

4.1 Objective Function

How to define the level of modularity depends on which level will lead most profit to the enterprise. Among different functions defined based on customers' requirements, many of them do not relate directly with customization; which means the modularity level for customization is not influenced by such functions. The requirements of customization that are related with the module partition include the following four factors: variety, upgrading ability, price and serviceability. Hence, the objective function for module partition considering customization can be expressed based weighted aggregation method (Tabucanon, 1988) as follows:

$$sl = f(u,v,p,s) = k_u u + k_v v + k_s s - k_p p \,, \qquad (2)$$

where

sl - the satisfying level of customers for the product;
u, v, p, s - measures on upgrading ability, variety level, price and serviceability;
k_u, k_v, k_p, k_s - specification weight of those four factors.

Different customers have different hobbies for the same product, so the weight distributions would be different. Assume there are a number of potential partition solutions or methods for a product, for the i^{th} partition method, its related attributes for upgrading ability, variety level, price and serviceability can be given as follows:

$$u_i = \frac{\max(mu_{ij})}{\max(u_k)}, \qquad (3)$$

$$v_i = \frac{\lg \prod_{j=1}^{n_i} mv_{ij}}{\lg \prod_{k=1}^{n} v_k}, \qquad (4)$$

$$p_i = \frac{(1 + r_a \frac{n_i}{n}) \left[\sum_{j=1}^{n_i} \left(Cdr_{ij} \cdot \sum_{k=1}^{n_{ij}} c_{jk} \right) \right]}{(1 + r_a) \sum_{k=1}^{n} c_k}, \qquad (5)$$

$$s_i = \frac{\sum_{k=1}^{n} f_k \cdot c_k}{\sum_{j=1}^{n_i} \left(f_{ij} \cdot Cdr_{ij} \cdot \sum_{k=1}^{n_{ij}} c_{jk} \right)}; \qquad (6)$$

where

mu_{ij}, mv_{ij}, Cdr_{ij}, f_{ij} - the upgrading frequency, number of variety, cost , ratio to total cost, order difficulty and failure possibility of the j^{th} module in the i^{th} partition method;
n_{ij} - the number of basic modules in the j^{th} advanced module;
u_k, c_k, f_k - the upgrading frequency, cost and failure possibility of the k^{th} basic module in the product;
n - the number of basic modules in the product;
n_i - the number of modules;
r_a - the rate of assembly cost to material cost;
c_{jk} - the cost of the k^{th} basic module in the j^{th} module.

Since there could be several choices for any basic module to meet the demand for specific customers, c_{jk} is the average cost of those modules which can be selected. Equation (4) uses logarithm to enlarge the range of v_i in equation so that it is able to attain similar weights with u_i, p_i and s_i. Cdr_{ij} is the discount factor for j^{th} advanced module. At the same time, that $Cdr_{ij} \cdot \sum_{k=1}^{n_{ij}} c_{jk}$ should not be less than the maximum value of c_{jk}.

mv_{ij} and f_{ij} are the maximums of varieties and failure possibilities of all the basic modules in the j^{th} module respectively; mu_{ij} is the minimum of upgrading ability of all the basic modules (u_k) in the j^{th} module.

Since the independent variables in equation (2) may have diverse dimensional range sizes, then normalization is needed. So, equation (2) can be changed as:

$$SL_i = k_u U_i + k_v V_i + k_s S_i - k_p P_i \,, \qquad (7)$$

where

$$U_i = \frac{u_i - u_{\min}}{u_{\max} - u_{\min}}, \quad V_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}},$$

$$P_i = \frac{p_i - p_{\min}}{p_{\max} - p_{\min}}, \quad S_i = \frac{s_i - s_{\min}}{s_{\max} - s_{\min}}. \qquad (8)$$

The scale of computing is determined as following. When the product merely contains one basic module, there is only one partition method; when two basic modules exist, two partition methods are applicable. For three basic modules, they can be potentially partitioned into five combination patterns as shown in Fig. 2. If there are many modules, although some of modules can not be combined into advanced modules due to certain engineering constraints, approaching the optimization of the objective function need numerous computations. Thus, this paper suggests Genetic Algorithm to seek optimization in a large computing scale.

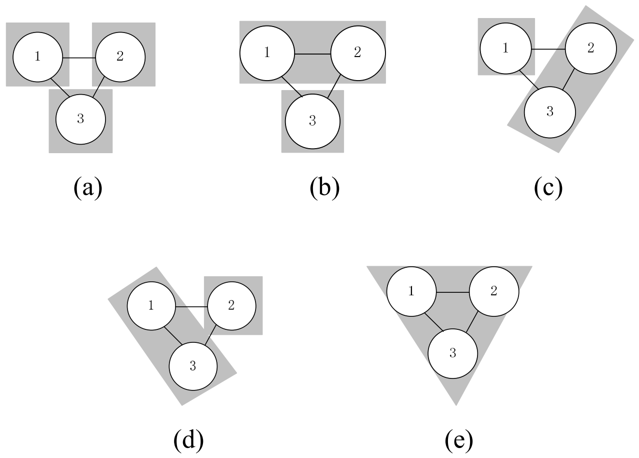

Figure 2 Partition methods of a product which includes three basic modules. Every shadow area represents a possible advanced module.

4.2 Optimization Algorithm

Genetic Algorithm (GA), an efficiently and effectively optimal technique, mimics the process of Darwinian to evolution to create populations from generation to generation (Goldberg, 1989, Kamrani and Gonzalez, 2003). By mixing and mutating existing chromosomes in the solution space, such an algorithm explores solutions to the combinational problems that would never be explored by regular optimization methods. The solution algorithm in this work follows the similar approach as follows:

(1) *Combination Matrix*. According to the connection matrix R of product, a combination matrix W_m is generated:

$$W_m = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1j} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2j} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ w_{i1} & w_{i2} & \dots & w_{ij} & \dots & w_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{nj} & \dots & w_{nn} \end{bmatrix}, \qquad (9)$$

where w_{ij}=0 or 1 and W_m is the mth partition method. When w_{ij} = 1, it represents that ith and jth basic modules are combined as in an advanced module, otherwise, they are not. This matrix is different from the above R, since each value in R is uniquely defined according to the product structure while matrix W_m reflects only the result of one partition methods. Matrix R is a constraint of W_m, and they are associated according to the same indices of rows and columns. Both of the two matrixes are symmetrical. When W_m is being generated, if r_{ij}=0, that w_{ij}=1, then such a partition method is invalid; so a new matrix will be generated.

(2) *Chromosomes Encoding*. To design a suitable chromosomes encoding scheme, the upper diagonal part of matrix W_m can be developed as potential solutions, $l = (w_{12}, w_{13}, \dots, w_{1n}, w_{23}, \dots, w_{2n}, \dots, w_{ij}, \dots, w_{(n-1)n})$. A feasible solution for the final objective function has to satisfy $w_{ij} \le r_{ij}$. According to equations (2)-(8), satisfying level to customers is then calculated, and parameters in the process are determined based on the real product. For the convenience of genetic algorithm which usually achieves the minimum of the population, so the objective function is inversed.

(3) *Crossing and Mutation*: *One Site Splice* method is used in this work to select a splice point randomly for each gene in each pair of parents. The first portions of the parents' genes are then exchanged. Figure 3 shows the crossover process in child generation. For two chromosomes F and M, they are divided into two portions. The vertical arrows represent the splice points; the latter portions of individual F and M generate the latter portions of chromosomes D and S respectively while they inherit the front portion from the parents directly.

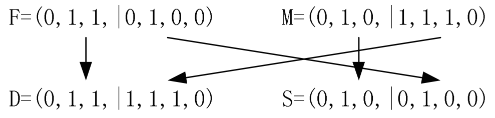

Figure 3 Crossover process

Mutation is a process of randomly disturbing genetic information to keep variety of population so to avoid premature termination of searching. In this paper, mutation rate is 0.2. Mutation operates at the bit level; when the bits are being copied from the parent to the child, there is a probability that each bit may become mutated. For instance:

Before mutation: $I = (1, 0, 1, 0, 0)$

After mutation: $I' = (1, 0, 1, 1, 0)$

Such crossover and mutation processes are repeated for each breeding cycle.

(4) *Comparison*. Newly generated matrixes are then tested by constraint, and if they satisfy conditions of the interconnection matrix, we compare ($-SL_1$) with ($-SL_2$). If ($-SL_1$) > ($-SL_2$), replace the chromosome of ($-SL_1$) with the one of ($-SL_2$), and record the value of ($-SL_2$). Otherwise, generate a new combination matrix till finding the minimum.

(5) *Replacement and Optimization*. Replacement is the last stage of any breeding cycles. This paper applies *Weakest Individual Replacement* (Kamrani and Gonzalez, 2003), which replaces the two weakest individuals in the population with the fitter children. Then, according to new chromosomes, new values of objective function will be achieved. When meeting the ending condition based on generation number or the convergence of the objective function, the computation stops. The minimum in the ending condition is the final result with selected parameters, and at the same time, we will gain the best partition method.

No. of Basic Module	1	2	3	4	5	6	7	8
Average cost c_k	25	8	40	30	10	8	100	80
Upgrading frequency u_k	1.5	2.0	8.5	8.0	1.0	1.5	6.7	7.0
Number of variety v_k	10	2	2	3	1	2	3	6
No. of Basic Module	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Average cost c_k	35	25	180	15	60	5	5	70
Upgrading frequency u_k	3.0	4.0	8.0	2.0	6.0	2.5	4.0	6.5
Number of variety v_k	10	2	8	6	2	6	1	6

5. A Case Study

A personal computer acts as the example case in this paper to illustrate how this optimization method works for partitioning modules in a product. Typically, PCs are designed and manufactured at a high modularity level; moreover, its variety and upgrading ability also strengthen its representation. A general PC is constituted with the basic modules as shown in Figure 4 together with the inter-connection graph and equation (11) with matrix.

According to the character of product, some useful data of optimization process should be given before computation. Table 1 describes the average costs, upgrading frequencies, and the number of varieties corresponding to the basic modules in PC.

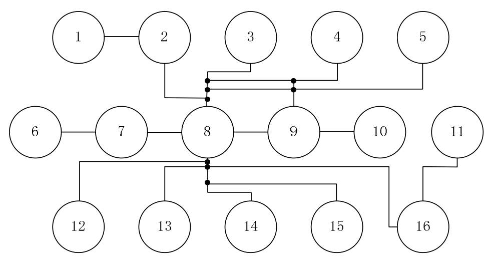

Figure 4 Product structure of PC with basic modules
1 Sound box; 2 Sound card; 3 Internal memory; 4 DVD-ROM; 5 Floppy disk drive; 6 Fan; 7 CPU; 8 Mother board; 9 Casing; 10 Power supply; 11 Monitor; 12 Keyboard; 13 Hard disk; 14 Mouse; 15 Network card; 16 Graphics card

In this example, f_k of all the basic modules are regarded as the same, i.e. 0.01. All *Cdr* will be 0.9; r_a = 0.2. The algorithm will stop after 100 generations. As shown in Fig. 4, there are 17 connections in this case; therefore, the number of variables in module partition is 17. Based on the above conditions, the initial population of algorithm is set as 20, crossover rate is 0.8, and mutation rate is 0.2. Since the algorithm generates random values as variables, we transform those larger than zero to 1, and others to zero. The algorithm is implemented with Matlab 7.0 on a computer with 1.4G Hz CPU and 512MB RAM.

After about one minute computation, the minimum of objective function and best individuals can be obtained. As shown in the lower half of Fig. 5, where the numbers on horizontal axis represent the linkage between different basic modules, i.e. 1-(1, 2), 2-(2, 8), 3-(3, 8), 4-(4, 8), 5-(4, 9), 6-(5, 8), 7-(5, 9), 8-(6, 7), 9-(7, 8), 10-(8, 9), 11-(9, 10), 12-(8, 12), 13-(8, 13), 14-(8, 14), 15-(8, 15), 16-(8, 16), and 17-(11, 16). For each of the values of these numbers, if it is positive, those related basic modules that are represented by the number are grouped into a common advanced module.

$$R = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ & & & & & & & & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & \cdots & & \cdots & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & 1 & 0 & 0 & 0 & 0 & 1 \\ & & & & & & & & & & & 1 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & & 1 & 0 & 0 & 0 \\ & & & & & & & & & & & & & 1 & 0 & 0 \\ & & & & & & & & & & & & & & 1 & 0 \\ & & & & & & & & & & & & & & & 1 \end{bmatrix} \quad (11)$$

Table 1 Characteristic attributes of basic modules

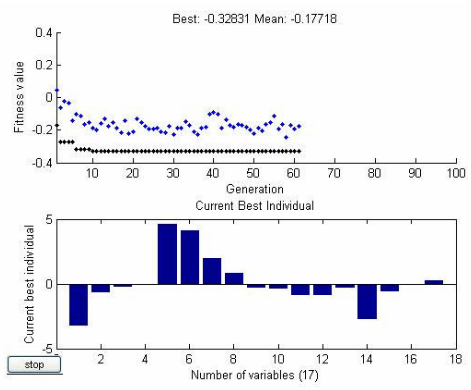

Figure 5 The optimization curve and best variables when *kv*=*ku*=*ks*=k_c=0.25

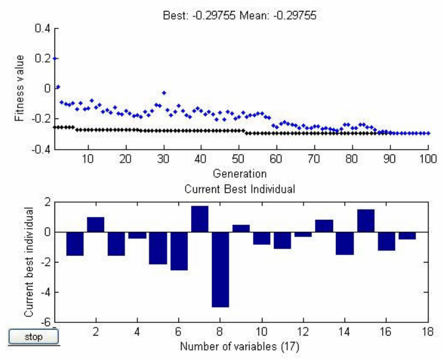

Figure 6 The optimization curve and best variables when k_u=0.1,k_v=0.2, k_c=0.3, k_s=0.4

After computation, when the coefficients k_v=k_u=k_c=k_s=0.25, minimum of the objective function is -0.328305918267453, x_5=x_6=x_7= x_8=x_{17}=1, and others are equal to zero. This result means basic modules 4, 5, 8 and 9 form an advanced module; basic modules 6 and 7, and 11 and 16 form other two separate advanced modules. Other basic modules should be manufactured or purchased independently. In addition, it can be appreciated that the designer could achieve different desired products to satisfy customers through selecting different parameters. When k_u=0.1, k_v=0.2, k_c=0.3, k_s=0.4 (Fig. 6), the resulted minimum is -0.297553099145687, and $x_2=x_7=x_9=x_{13}=x_{15}=1$. Then basic modules 2, 7, 8, 13 and 15 combine into an advanced module; and basic modules 5 and 9 combine another.

6. Conclusions and Future Research

The new demand on mass customization in manufacturing industry will have a huge influence on the degree of modularization. Customers need most suitable products. This paper proposes that since modules in a product can be divided into basic modules, sub-modules and advanced modules, there maybe numerous potential module partitions considering customer oriented design requirements, such as product variety, upgrading ability, cost, serviceability, etc. To a specific group of customers, there must be a best suitable partition. This paper seeks the module partition optimization solution especially for customization with a genetic algorithm. The idea and method of which could help designers to develop more customized products.

Several further research works should be fulfilled in the future. When defining the objective function of modularity, the deep research on which requirements of customers should be considered and how they are influenced by different module partitions, will increase the accuracy of final selection by designers in a large degree. The inter-connection natures and strengths among different modules, e.g. positive or negative, strong or weak, are also worth further study. Moreover, as we all know, genetic algorithm has its own disadvantages such as premature termination on searching, which means we may attain a local optimization rather than a global one. Therefore, an improved or new algorithm to optimize the objective function can be expected.

References

1. Pine B. J. II., 1993, “Mass Customization-the New Frontier in Business Competition”, Boeton, Harvard Business Press.

2. Silveira G. D., Borenstein D., Fogliatto F. S., 2001, “Mass customization: Literature review and research directions”, International Journal Production Economics v 72, p 1-13.

3. Gaimon C., Singhal V., 1992, “Flexibility and the choice of manufacturing facilities under short product life cycles”, European Journal of Operational Research, v 60, n 2, p 211-223.

4. Baldwin C. Y., Clark K. B., 2000, “Managing in an Age of Modularity”, Harvard Business Review, p 84-93.

5. Fixson S.K., 2002, “The Multiple Faces of Modularity-An Analysis of a Product Concept for Assembled Hardware Products”, MIT, Cambridge MA, Working Paper,

6. Kusiak A., Huang C. C., 1998, “Modularity in Design of Products and Systems”, IEEE Transactions on System, Man, and Cybernetics- Part A: System and Humans, v 28, n 1, p 66-77.

7. Kamrani A., Gonzalez R., 2003, “A genetic Algorithm-based Solution Methodology for Modular Design”, Journal of Intelligent Manufacturing, v 14, p 599-616.

8. Kreng V. B., Lee T. P., 2004, “Modular product design with grouping genetic algorithm-a case study”, Computers and Industrial Engineering, v 46, p 443-460.

9. Albano, L.D., Suh, N. P., 1992, “Axiomatic approach to structural design”, Research in Engineering Design, v 4, p 171-183.

10. Erens F., Verhulst K., 1997, “Architecture for Product Families”, Computer in Industry, v 33, p 165-178.

11. Chen, G., Ma, Y.-S., Thimm, G. and Tang, S.-H., “Associations in a Unified Feature Modeling Scheme”, ASME Transactions Journal of Computing & Information Science in Engineering, v 6, n 2, 2006.

12. Jiao, Roger J.X., Simpson, T.W. and Siddique, Z., 2006, “Product Family Design and Platform-based Product Development: A State-of-the-Art Review”, Accepted by Journal of Intelligent Manufacturing.

13. Ulrich K., 1995, “The role of product architecture in the manufacturing firm”, Research Policy, v 24, p 419-440.

14. Kusiak A., Huang C. C., 1996, “Development of Modular Products”, IEEE Transactions on Components, Packaging, and Manufacturing Technology-Part A, v 19, n 4, p 523-538.

15. Mikkola J. H., 2001, “Modularity and Interface Management of Product Architectures”, Portland International Conference on Management of Engineering and Technology, Proceedings, v 1, p 599-609.

16. Tabucanon M. T., 1988, “Multiple Criteria Decision Making in Industry”, Elsevier, Amsterdam.

17. Goldberg D. E., 1989, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison-Wesley.