Hierarchical Clustering Approach for Product Variety Management

POOYA DAIE

A Thesis

in the Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Industrial Engineering) at

Concordia University

Montreal, Quebec, Canada

September 2015

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Pooya Daie**

Entitled: **Hierarchical Clustering Approach for Product Variety Management**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____ Chair
Dr.

____ External Examiner
Dr. Fazle Baki

____ External to Program
Dr. Navneet Vidyrathi

____ Examiner
Dr. Onur Kuzgunkaya

____ Examiner
Dr. Mingyuan Chen

____ Thesis Supervisor
Dr. Simon Li

____ Thesis Co-Supervisor
Dr. Ali Akgunduz

Approved by __________Dr. Ali Dolatabadi____________________________
Chair of Department or Graduate Program Director

September, 2015 ____
Dr. , Dean
Faculty of Engineering and Computer Science

ABSTRACT

Hierarchical Clustering Approach for Product Variety Management

Pooya Daie,

Concordia University, 2015

Continually evolving customer's needs has contributed to an increase in demand for product variety over the recent decades. Proliferation of product variants affects different aspects of product life cycle which increases the complexity of managing product variety. In this context, the notion of grouping and classification based on similarity within a family of product is the key in managing product variety. This research proposes hierarchical clustering as solution approach that is intuitively relevant and it focuses on progressively grouping the elements that share high similarity with each other.

In this research, three types of product variety-related problems are investigated. The first problem concerns with designing product architecture in a way to support product variety. Design Structure Matrix (DSM) is used to visualize product architecture and to develop a new matrix-based clustering approach based on hierarchical cluster analysis. The challenge is that there are numerous possible product architectures even for a product with few components. One unique advantage of the proposed method lies in supporting "overlapping components" which is not directly addressed by the conventional techniques in cluster analysis.

The second problem focuses on structuring supply chain network in case of product variety that indicates the precedence orders of suppliers and sub-assemblers. The challenge is that the number of possible structures of supply chain network grows dramatically with the increase in the number of product variants. The solution approach is based on hierarchical clustering, in which the tree structure is applied to construct the supply chain network. The core technique is to investigate the coupling values between the module variants and characterizing the grouping condition in the structuring process.

The third problem is to develop semi-finished products to reduce production costs. The challenge is that the possible solution space can increase exponentially with increase in the number of elements (e.g. components) in the problems. In the solution approach, the basic information of product variety is captured in a matrix format, specifying the component requirements for each product variant. Then, hierarchical clustering is applied over the components with the consideration of demands. The key stage is similarity analysis, in which problem-specific information can be incorporated in the clustering process.

In summary, the proposed method can be a practical tool for tackling product variety-related problems. It yields good quality results in a limited time. Thus, it can be used to obtain better results than other algorithms when the amount of time available to perform the task is limited.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr. Simon Li for all of his supports, valuable guidance and commitments during the course of this research. This work was possible only because of your unconditional support, understanding, and immense knowledge. Thank you very much for providing me with the opportunity to learn.

I would like to extend my appreciation to my co-supervisor Dr. Ali Akgunduz for his invaluable supports and advices.

I wish also to thank my committee members for their precious time, brilliant comments and insightful suggestions. Also thanks to the faculty members and staffs of MIE who helped me whenever I approached them. A special acknowledgement goes to all my amazing friends in Concordia University.

I would like to extend my deepest gratitude and appreciation to my adored parents and my great brothers for all of their supports and ultimate kindnesses. Their prayers and love during this times have encouraged me to do my best to achieve my goal. Words cannot express my appreciation to them for their ultimate supports and goodwill. I have saved the last words of this acknowledgment for my brilliant wife. I would like to thank her for her supports, care, patience and faith in me.

Thesis Format

This thesis has been prepared in “Manuscript-based” format. All the contents that are represented in this thesis either have been published, accepted for publication or submitted for publication in the form of journal articles in which the student is the first author. All the papers presented in this thesis were co-authored and reviewed prior to submission for publication by my supervisor Dr. Simon Li. All of the remaining work and manuscript preparation were performed by the author of this thesis.

Contents

ABSTRACT iii

Thesis Format vi

List of Figures xi

List of Tables xiv

Chapter 1. Introduction 1

1.1. Product Variety Management 1

1.2. Variety Enablers and Techniques 3

1.2.1 Modular product architecture 3

1.2.2 Postponement strategy 5

1.2.3 Developing semi-finished products (vanilla box) 8

1.3. Research Motivation 9

1.4. Research Objectives 11

1.5. Organization of the Thesis 12

Chapter 2. Developing Modular Architectures in Support of Product Variety 14

2.1. Introduction 15

2.2. Literature Review 17

2.3. Product Architecture and Design Structure Matrix 18

2.3.1 Matrix representation 18

2.3.2 Minimum description length 21

2.4. Methodology 22

2.4.1 Matrix-based hierarchical clustering (MHC) 22

2.4.2 Procedure for DSM clustering 27

2.5. Application and Comparison Study 35

2.5.1 GM powertrain (GMPT) 35

2.5.2 Gas turbine generator (GAS) 38

2.5.3 Comparison with GA as a methodical approach 41

2.6. Conclusions 43

2.7. Appendix 44

2.8. Author's Notes and Significance of Paper to Thesis 46

Chapter 3. Hierarchical Clustering for Structuring Supply Chain Network in Case of Product Variety 47

3.1. Introduction 48

3.2. Background and Related Works 50

3.2.1. Supply chain network & postponement 50

3.2.2. Computational complexity of supply chain network 54

3.3. Problem Statement and Complexity 56

3.3.1. Problem description 56

3.3.2. Complexity model of a supply chain structure 60

3.4. Methodology 61

3.4.1. Concept of coupling in supply chain network 61

3.4.2. Methodical procedure 62

3.5. Numerical Examples 68

3.5.1. Discussion of Case #1 and Case #2 68

3.5.2. Discussion of Case #3 and Case #4 71

3.5.3. Discussion of Case #5 73

3.6. Closing Remarks 76

3.7. Author's Notes and Significance of Paper to Thesis 78

Chapter 4. Managing Product Variety through Configuration of Pre-assembled Vanilla Boxes Using Hierarchical Clustering 79

4.1. Introduction 80

4.2. Literature Review 81

4.3. Background 83

4.3.1. Vanilla box configuration problem 83

4.3.2. Hierarchical clustering 84

4.3.3. Remarks 85

4.4. Hierarchical Clustering Methodology 85

4.4.1. Similarity analysis 86

4.4.2. Tree construction 87

4.4.3. Tree-based analysis 89

4.5. Numerical Examples 90

4.5.1. Vanilla box for drill design 90

4.5.2. Incorporating demand information in VBC problems 93

4.5.3. Remarks on the method’s utility 99

4.6. Conclusion 100

4.7. Author’s Notes and Significance of Paper to Thesis 101

Chapter 5. Conclusions 102

5.1. Summary 102

5.2. Concluding Remarks on the Utility of Hierarchical Clustering 103

5.3. Contributions 106

5.4. Suggestion for Future Research 108

References 110

Appendix A. Configuration of assembly supply chain using hierarchical cluster analysis 120

List of Figures

Figure 1-1 Product architecting problem 4

Figure 1-2 Postponement vs Early differentiation 7

Figure 2-1 DSM clustering: input and output 19

Figure 2-2 Schematic view of product architecture with a bus and one interactive component .. 21

Figure 2-3 Similarity matrices 24

Figure 2-4 Algorithms for tree construction and tree node sequencing 26

Figure 2-5 Resulting matrix after sorting analysis 27

Figure 2-6 Approximate structure of the sample matrix 31

Figure 2-7 Concept of partition point in a diagonal matrix 32

Figure 2-8 Tree cutting and number of modules 33

Figure 2-9 Preliminary modules and final structured matrix 35

Figure 2-10 Approximate structure (GMPT) 36

Figure 2-11 Final structured matrix (GMPT) 37

Figure 2-12 Sorted matrix after applying MHC for the first time (GAS) 39

Figure 2-13 Final structured matrix (GAS) 40

Figure 2-14 Manually clustered GMPT 44

Figure 2-15 Clustering arrangement by GA for GMPT 44

Figure 2-16 Manually clustered GAS 45

Figure 2-17 Clustering arrangement by GA for GAS 45

Figure 3-1 Modular supply chain vs Non-modular supply chain 52

Figure 3-2 General supply chain network 54

Figure 3-3 Illustration of modules and product variants 57

Figure 3-4 Mix ratio of product variants 58
Figure 3-5 A sample structure of supply chain network 59
Figure 3-6 Product variety form 63
Figure 3-7 Coupling matrix of module options 64
Figure 3-8 Adjusted coupling matrix 65
Figure 3-9 Coupling matrix between modules 65
Figure 3-10 Tree-based analysis 66
Figure 3-11 Two possible structures of supply chain network 66
Figure 3-12 Product variety forms of Case #1 and Case #2 69
Figure 3-13 Tree result and supply chain network of printer example (Case #1) 69
Figure 3-14 Tree result and supply chain network of case #2 70
Figure 3-15 Product variety forms of Case #3 and Case #4 71
Figure 3-16 Tree construction and supply chain network of Case #3 72
Figure 3-17 Tree construction and supply chain network of Case #4 72
Figure 3-18 Product variety form of Case #5 74
Figure 3-19 Supply chain network of Case #5 75
Figure 4-1 Example of product variants 84
Figure 4-2 sample product variety matrix and its relevant similarity matrix 84
Figure 4-3 Tree construction and updated similarity matrix 88
Figure 4-4 Product variety matrix of cordless drills 90
Figure 4-5 Tree structure of cordless drills 91
Figure 4-6 Cordless drills 92
Figure 4-7 Product variety matrix in example 2 93

Figure 4-8 Tree structure and platform solutions for Scenario 1 96

Figure 4-9 Tree structure of Scenarios 2 to 5 97

List of Tables

Table 1-1 Input information and solution outputs of product variety problems 10

Table 2-1 Partition points and similarity values 33

Table 2-2 Start and end points of preliminary modules and final structure matrix 35

Table 2-3 Comparison results of the GMPT example 37

Table 2-4 Comparison results of the GAS example 41

Table 3-1 Complexity measure for Case #1 and Case #2 71

Table 3-2 Exhaustive comparison of complexity measures in Case #3 and Case #4 73

Table 4-1 Platform solutions and their costs in Scenarios 2, 3, 4 and 5 98

Table 5-1 Comparison of the proposed solution method with existing approaches 104

1. Introduction

1.1. Product Variety Management

Nowadays, there is an increasing emphasis on offering product variety. For example, BMW claims that "every vehicle that rolls of the belt is unique" and the number of possible variations in BMW 7 series could reach 10^{17} (Hu et al., 2008). Product variety is available regardless of the complexity of products, which can be as simple as a mug or as complex as an airplane. Variety can be defined as a number or collection of different types of a specific class of the same generic kind (ElMaraghy et al., 2013). The variety concept applies where the provider aims to gain economic benefits by offering a wider range of products with the possibility of customization and even personalization by customers. The increase in requiring variety has several reasons including arising demands of customers for new products with new functions and features, different geographical requirements and gaining more market share by companies (ElMaraghy et al., 2013, Hu et al., 2011).

Offering product variety to customers can be a potential source of benefits in terms of expanding the market, increase sales and revenues. However, proliferation of product variants complicates related design, manufacturing and supply chain activities that may increase costs and reduce profits (Johnson, 2003). Thus, its positive outcomes are not guaranteed unless variety is well managed across all stages of product lifecycle from design to manufacturing and supply chain.

Producing different variants of products imposes increased complexity level on a firm to design, produce and manage those products. In terms of product design, offering product variety can complicate the design process by increasing the number of designs that designers deal with. In terms of manufacturing, it could be the reason for increased costs and lead-times due to the setup cost/time required for manufacturing different variants of products. Also, it could be the cause for increased inventory costs in supply chain as more inventory of finished goods should be stored in order to respond to different needs of customers. Thus, it is imperative to employ variety-oriented approaches for managing product variety. In this way, different approaches including product modularity, platform-based product development and postponement have been proposed in order to mitigate some of the negative effects of the increased product variety (AlGeddawy and ElMaraghy, 2012, ElMaraghy et al., 2013, Da Cunha Reis et al., 2013).

Variety management strategies and enablers can be implemented to different stages of product life cycle. Also, its granularity ranges from parts and modules to products. In general, one of the important objectives of variety management strategies is to achieve mass customization to reduce and manage the complexity and its associated costs that are induced by product variety. Mass customization that was first coined by Davis (1989) in late 1980s and later developed by Pine and Victor (1993) relates to the ability of the firm to produce customized products in high volumes with near mass production costs and efficiency (Da Silveira et al., 2001). In this way, the notion of grouping and classification based on similarity and commonality within a family of product is the key success factor in achieving mass customization and managing variety. The principle of "not re-inventing the wheel" whenever a new variant is offered is the basis for any product variety management approach (ElMaraghy et al., 2013).

In the literature, different techniques and solution approaches have been used for tacking problems incurred by offering product variety. This research aims to find a solution approach based on the notion of similarity between components/product variants to solve product variety-related problems. The proposed solution approach is applied to three types of product variety-related problems. Some challenges of offering product variety are highlighted and solutions are suggested that can be helpful for managers on making decisions regarding product variety issues.

1.2. Variety Enablers and Techniques

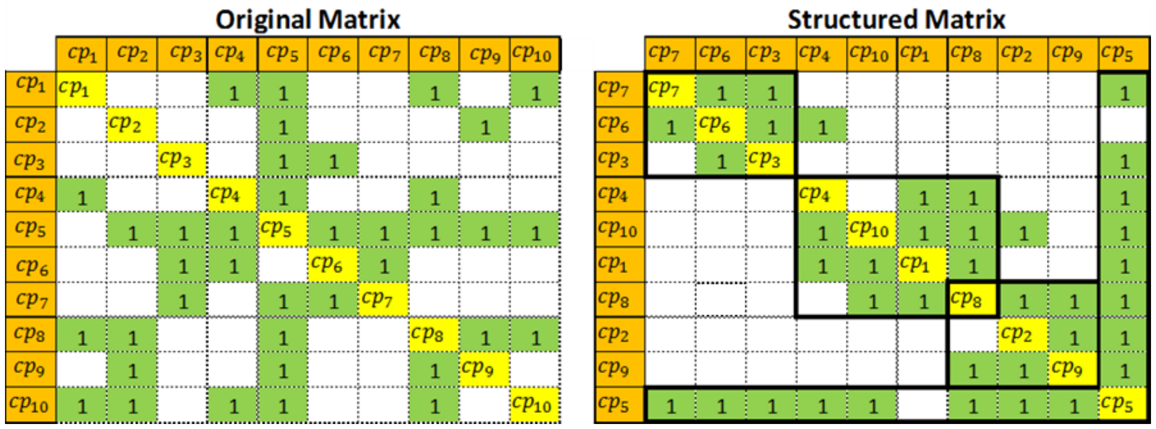
Product variety enablers are techniques, methodologies and strategies that are employed to manage product variety and support mass customization. This thesis focuses on modular product design, postponement and platform-based manufacturing as the most significant enablers of mass customization for managing product variety. Specifically three types of product variety-related problems including identifying modular product architecture, constructing structure of supply chain network and configuration of pre-assembled vanilla boxes are investigated.

1.2.1 Modular product architecture

Research studies show that about 80% of the product costs are determined by the design of the product (Mikkola, 2007). Thus, companies rely on product design as a means to achieve mass customization. Modularity is one of the main enablers of mass customization. In modular product architecture design, a product is structured via several modules with standardized interfaces that allow swappable features to deliver product variants. Product variety can be achieved by substitution of the modules based on customer demand. In this way, modular product architecture helps to increase product variety through reducing time to market and costs (Wang et al., 2014).

In the context of product architecture, the matrix-based product architecting problem is to transform the original matrix to structured matrix on which modular product architecture can be identified (Figure 1-1). In this way, offering product variety is enabled through standardizing the interfaces between modules of the product. For instance, in Figure 1-1, the first module ($\{cp_7$,

$cp_6, cp_3\}$) can be replaced with another version of that module if all the interfaces between its components and other modules are standardized. That is, it is crucial to standardize interface between cp_6 and cp_4 in order to facilitate substitution of this module for offering product variety. However, standardizing interfaces between modules always come with extra costs. Therefore, in support of product variety, product architecture design aims to minimize such interfaces between the modules of product.



Original Matrix

	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	cp_8	cp_9	cp_{10}
cp_1	cp_1			1	1			1		1
cp_2		cp_2			1				1	
cp_3			cp_3		1	1				
cp_4	1			cp_4	1			1		
cp_5		1	1	1	cp_5	1	1	1	1	1
cp_6			1	1		cp_6	1			
cp_7			1		1	1	cp_7			
cp_8	1	1			1			cp_8	1	1
cp_9		1			1			1	cp_9	
cp_{10}	1	1		1	1			1		cp_{10}

Structured Matrix

	cp_7	cp_6	cp_3	cp_4	cp_{10}	cp_1	cp_8	cp_2	cp_9	cp_5
cp_7	cp_7	1	1							1
cp_6	1	cp_6	1	1						
cp_3		1	cp_3							1
cp_4				cp_4		1	1			1
cp_{10}				1	cp_{10}	1	1	1		1
cp_1				1	1	cp_1	1			1
cp_8					1	1	cp_8	1	1	1
cp_2								cp_2	1	1
cp_9							1	1	cp_9	1
cp_5	1	1	1	1	1		1	1	1	cp_5

Figure 1-1 Product architecting problem

Designing product architecture is a challenging task which is a key activity in the engineering design of modular products. That is, to the factorial of the number of product components, we can have different arrangements of the components on the row/column heading of the DSM and consequently different product architectures. Since each sequence of the components on the row/column headings of the matrix implies a unique design of product architecture, the number of possible product architectures can be increased dramatically with the increase in the number of product component which makes the combinatorial nature of the problem challenging.

In this context, the first part of this research (Chapter 2) focuses on developing modular product architecture. It captures the dependencies between components of the product to identify the structure of product in terms of composition of the modules and identification of the interfaces between modules. Matrix representation of product architecture is employed to describe and visualize product architecture. Accordingly, a new matrix-based clustering approach is developed to identify sub-systems or modules of the product to support product variety.

1.2.2 Postponement strategy

The concept of postponement has a long history both in academic literature and implementation practices (Pagh and Cooper, 1998). It is an operational concept that deals with the delay of product differentiation point until customer's orders have been received (Kim, 2014). Figure 1-2 shows concept of postponement versus early differentiation. Postponement enables inventory cost reduction as there is less randomness in demand for base modules (e.g., platform) than demand for differentiating modules of products. The growth of interests in the postponement by firms is resulted from the increased demand for customized product by customers (Kisperska-Moron and Swierczek, 2011).

In terms of application, modularity and high commonality of the modules are important characteristics of postponement. For instance, modular product architecture enables manufacturers to provide variety of end products on the basis of few generic platforms (Yang and Yang, 2010). This concept is embraced in postponement strategy to delay the point at which differentiating modules join generic modules in order to reduce inventory costs. For example, in Figure 1-2, $\{md_{1,1}\ md_{2,1}\}$ is a generic sub-assembly that are used in all product variants. Therefore, by postponing differentiating modules ($\{md_{3,1}, md_{3,2}, md_{3,3}\}$) till finalizing the order from the customer, some inventory costs can be saved.

Nevertheless, implementing postponement in a pre-defined supply chain network where the position and product of the suppliers and sub-assemblers are already fixed will limit its effectiveness. Therefore, unlike the supply chain configuration problems in literature that often assume the "fixed" supply chain structure, termed as "generic supply chain network" (Huang et al., 2005), which is concerned with the optimal selection of suppliers subject to various considerations like lead-time and cost (Graves and Willems, 2005); this research aims to construct the supply chain network that indicates the precedence orders of suppliers and sub-assemblers based on the product variety information.

The fundamental challenge of such structuring problem is that the possible number of structures are numerous even for a small number of elements (e.g. suppliers or modules). That is, the number of possible supply chain network with n elements is between 2^{n-1} and $n!$ (Webbink and Hu, 2005). Thus, generating all possible supply chain network structures to select the most appropriate structure among them based on the various objectives (e.g. complexity or cost) is a challenging task due to the computational complexity incurred by numerous possible configurations.

In this way, the second part of this research (Chapter 3) addresses the issues of applying postponement strategy on structuring supply chain network in case of product variety. A methodical procedure is developed to construct the supply chain network in terms of precedence orders of suppliers and sub-assemblers in a way to support postponement and reduce complexity of the network. The structuring problem specifically considers two pieces of product variety information: (1) the number of different types of product variants to be offered, and (2) the demands of various product variants. The research results are relevant for supply chain managers who need to make decisions in different supply chain practices such as postponement, procurement policy and supplier selection. It provides insights to enhance supply chain performance through considering postponement strategy in the structuring of supply chain network.

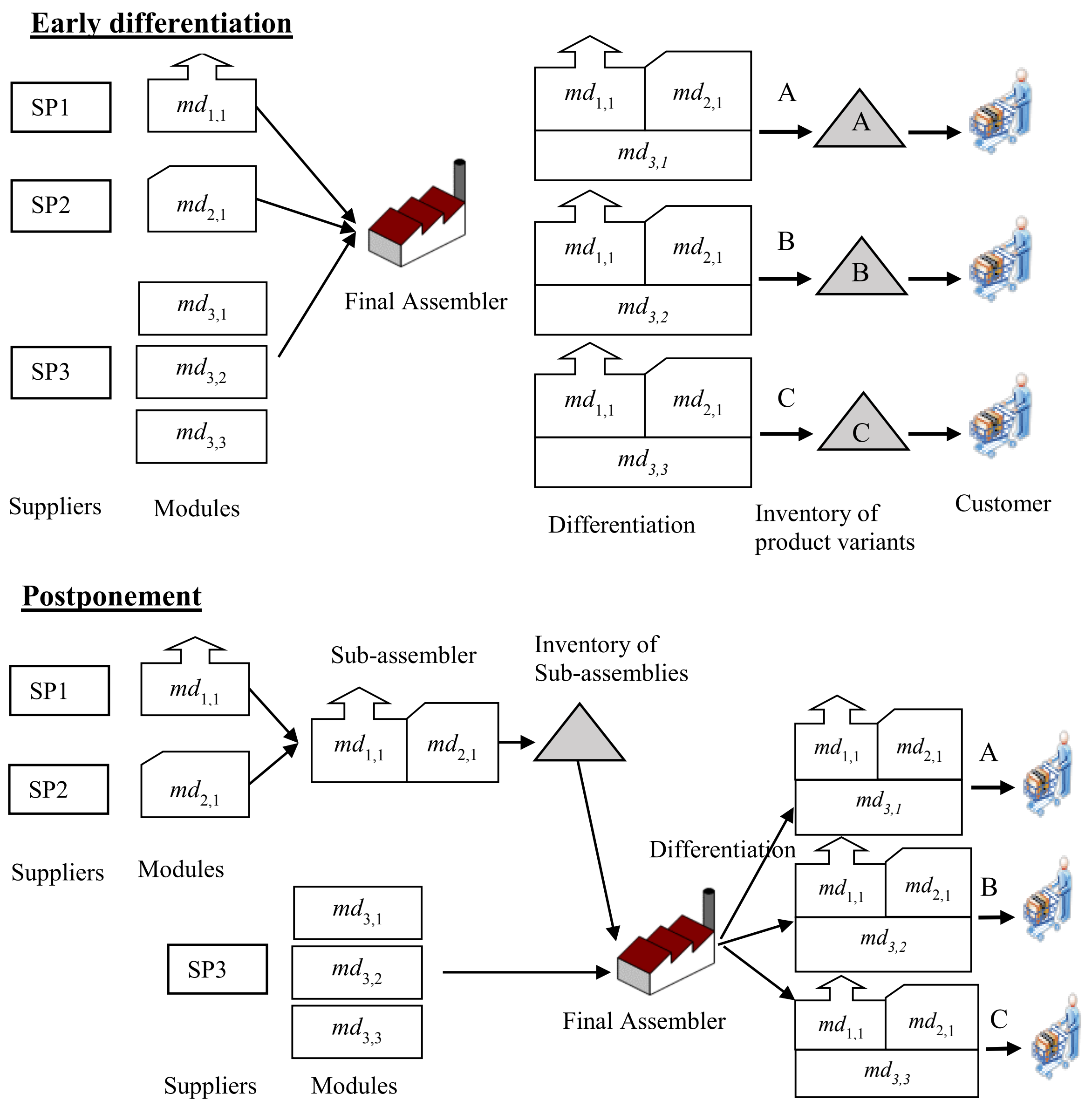


Figure 1-2 Postponement vs Early differentiation

1.2.3 Developing semi-finished products (Vanilla Box)

Components sharing in terms of platforming has become an important way of cost-saving across different industries. For example, Volkswagen's MQB platform is used in production line of VW Golf, Audi A3 and Skoda Octavia (Simpson et al., 2014). The increase in use of platform-based manufacturing is in response to the market demand for product variety. For instance, nowadays, consumers have come to expect different price ranges for a hand drill to choose from (Halman et al., 2003). This variety has a direct impact on the firm's performance in terms of increasing manufacturing complexity to support this variety level. As an example, a car can have about five million possible variants when considering all the possible options and combinations (Cameron, 2011).

Platform-based manufacturing is a strategy for providing variety to the market at a reduced costs. It can provide significant competitive advantages if executed well in the company. Firms can cut costs by 30% and reduce lead-time by 50% by employing platform-base manufacturing (Simpson et al., 2014). However, gaining such competitive advantage is not easy and the list of companies that have attempted and failed to build a platform is long (Boas et al., 2013).

The platform concept have been used for different objectives such as cost reduction and reducing time-to-market (Simpson et al., 2001, Simpson, 2004). The basic idea behind platform-based manufacturing is to reduce assembly operation in terms of setup cost/time by developing some semi-finished forms of the product. These semi-finished forms of product are called vanilla boxes (Swaminathan and Tayur, 1998). In the context of mass customization, if this vanilla box can be mass produced with a lower costs, it is expected to decrease the production costs and simplify the assembly operations. That is, if vanilla boxes are designed in a way that they can be used in the production process of many product variants with minimum changes (in terms of adding or removing components), it will outweigh the setup cost that is needed for forming vanilla boxes and would be beneficial for reducing production costs.

In this context, the combinatorial nature of the problem makes the solution space grow dramatically specially with the increase in number of product component. In practice, as each vanilla box incurs a setup cost, it may not be economical to have many small vanilla boxes. Alternately, a large vanilla box can make it less useful for most product variants. Thus, finding the "balanced" configuration of vanilla boxes is not a trivial problem.

The Vanilla Box Configuration (VBC) problem is addressed in the third part of this research in Chapter 4. It concerns with finding subsets of components (i.e. vanilla boxes) that are common to some product variants in order to reduce costs. In this research, hierarchical clustering is proposed as a practical tool for solving the VBC problem in the management of product variety.

1.3. Research Motivation

In the investigation of the product variety enablers in Section 1.2, it is found that these problems often require some grouping (or formation) decisions that implicate certain structural outputs. In product modularization, the formation of modules needs the grouping of components in order to minimize the interfaces. The postponement strategy in structuring supply chain network seeks for the point of differentiation that determines the formation of sub-assemblies. In the Vanilla Box Configuration (VBC), it is intended to reduce the production cost via formation of semi-finished products that can be produced with a relatively high volume.

Table 1.1 summarizes the input information and solution outputs of three types of product variety problems. Since these problems involve some grouping decisions, the traditional optimization formalism usually requires integer variables and certain constraints to control the numerical procedures for structural solution outputs. At the same time, the possible solution space can increase exponentially with the increase of the number of elements (e.g. components) in the problems. Thus, the global optimality is not commonly claimed in the existing literature and the current research.

Table 1-1 Input information and solution outputs of product variety problems

	Input information	**Solution outputs**
Product Modularization	Physical dependency of components	Modules and interfaces
Structuring Supply Chain Network (postponement)	Product variants and their demands	Sub-assemblies (showing points of differentiation)
Vanilla Box Configuration	Product variants and their production costs	Semi-finished products

Given the computational challenges, the research motivation is to investigate the application of hierarchical clustering for the product variety problems. First of all, hierarchical clustering is intuitively relevant as it focuses on the successive grouping of elements and does not require the number of groups as the input. Its technical contents are rigorous and have been well developed in literature (Everitt et al., 2011). Computationally, hierarchical clustering takes the non-exhaustive strategy so that it has a relatively short computing time (i.e., a matter of seconds or few minutes in most cases) but does not guarantee global optimality.

Generally, using hierarchical clustering as the solution approach for product variety management problems can be described with the following characteristics:

- Hierarchical clustering is a relatively simple solution method. This characteristic is both reflected in terms of required computational times for solving problems using hierarchical clustering and also less problem formulation difficulties in terms of defining constraints and parameters.

- Hierarchical clustering consists of independent phases that change in one phase do not affects the procedure of other phases. This characteristic of hierarchical clustering makes it quite flexible solution approach in terms of adding a new parameter into consideration. That is, considering a new parameter in one phase do not affects the procedure of other phases and therefore new factors can easily be considered in the solution process.

- As hierarchical clustering is not an exhaustive search algorithm, the global optimality cannot be claimed. However, considering the three types of product variety problems mentioned in this research, optimal solutions are not easy to obtain anyway as the size of the problem grows.

In this research, it is considered that hierarchical clustering can offer simplicity and flexibility in solving the product variety problems. Through the literature comparison, one research inquiry is to examine the performance of the hierarchical clustering in solving product variety problems. The thesis is intended to demonstrate that the proposed solution approach is effective for solving such combinatorial problems in the context of managing product variety.

1.4. Research Objectives

This manuscript-based thesis consists of the three following journal papers that details of each paper and applied methodology will be discussed in the relevant chapters.

- Paper 1: *Developing modular architectures in support of product variety*

It aims to identify modular structure of the product in terms of different sets of components (bus, interactive and module components). Matrix representation of product architecture is used to describe and visualize system architecture and a matrix-based clustering method is developed to identify modular architecture of the product.

- Paper 2: *Hierarchical clustering for structuring supply chain network in case of product variety*

The objective of this paper is to construct the structure of supply chain network considering product variety information in a way to reduce the complexity of supply chain network. It proposes hierarchical clustering as the numerical tool to facilitate the structuring process of supply chain network.

- Paper 3: *Managing product variety through configuration of pre-assembled vanilla boxes using hierarchical clustering*

The objective of this paper is to develop vanilla boxes based on the information of product variants in order to facilitate the process of offering product variety in terms of reducing cost. Hierarchical clustering approach is employed for solving VBC problem in the management of product variety.

1.5. Organization of the Thesis

This sub-section outlines the layout of this manuscript-based thesis. This thesis consists of five chapters and an appendix related to Chapter 3. Chapter 1 provides an introduction about offering product variety and relevant challenges for managing product variety. The key enablers of mass customizations which are modular product architecture, postponement and developing semi-finished products are briefly discussed. Chapter 2 is dedicated to the study of developing modular product architecture that supports product variety. Three different sets of components are defined and a matrix-based clustering method is developed for identifying product architecture that determines these sets of components in a product. In Chapter 3, a method is proposed for constructing structure of supply chain network that supports postponement strategy and reduce the complexity of supply chain network. Chapter 4 focuses on developing common semi-finished products to facilitate production of product variants in terms of manufacturing and assembly process. Chapter 5 summarizes this research by providing conclusions, contributions and recommendations for future works. Appendix A is related to Chapter 3 that focuses on configuration of assembly supply chain using hierarchical clustering. Moreover, in order to maintain the integrity of the thesis, at the end of each chapter, connecting texts that provides bridges to the next chapter is provided.

2. Developing Modular Architectures in Support of Product Variety

ABSTRACT

Product architecture can influence different aspects of product lifecycle including manufacturing, assembly and supply chain. The purpose of this paper is to employ hierarchical cluster analysis for identifying product architecture to support product variety. Design Structure Matrix (DSM) is used to visualize and analyze product architecture and develop the required building blocks for identifying product modules. The proposed method for DSM clustering consists of three phases. The first phase is component filtering to identify components that cannot be classified in a module. The second phase is approximate structure formation that preliminarily organizes similar components close to each other in a matrix format. The third phase is partitioning analysis that finalizes the modules' boundary to yield the structured matrix as the solution of DSM clustering. To examine the solution's quality, minimum description length from literature is used as the comparable objective. Then, the proposed method is demonstrated via two literature examples and compared with the solutions by the manual and genetic algorithm approaches. One unique advantage of the proposed method lies in its flexibility that allows domain experts to provide their contextual judgment in the formation of the structured matrix as the final solution.

2.1. Introduction

To compete in the market share, one trend of product development is to provide more choices to customers via customization that manufacturers produce multiple variants of the same product with different features and prices. While this trend is appealing to customers, it incurs the challenges in view of design, manufacturing and supply chain. In this context, one common practice is to employ the concept of product architecture (Ulrich, 1995, Eppinger and Browning, 2012, Simpson et al., 2001). Generally, product architecture can be described with the following characteristics.

- ❖ The relations between components of a product are captured via their connections in view of energy, material and information flows.
- ❖ Several components can be grouped to form a module. The relations between modules are often termed as "interfaces". Each module is expected to deliver some specific functions of a product.
- ❖ To offer product variety, it is intended to minimize and standardize the interfaces of modules. In this way, the functions (or features) of the product can be modified by changing the modules with the compatible interfaces.

Product architecture plays a significant role to manage product variety in view of design and supply chain. In design, the formation of modules affects the architectural details of the product. At the same time, the standardization of interfaces often restricts some design freedom and incurs extra costs. In supply chain, as some identical modules can be shared by multiple product variants, their demands tend to be stable and can be produced with higher volume. Then, cost saving is possible via delayed differentiation that keeps uncertain “differentiating” modules to be produced and implemented closer at the user side (Eppinger and Browning, 2012).

One general problem of product architecture is how to group the components so that the product can be properly “architected”. To address this problem systematically, it is common to use graphs and matrices to represent the relational information of components (Pimmler and Eppinger, 1994). This paper uses product Design Structure Matrix (DSM) to represent product architecture (Browning, 2001). Generally, product DSM captures the relations between any two components of the product. Then, the grouped components can be rearranged in the DSM that shows the formed modules and their interfaces. In literature, this problem is also referred to DSM clustering (Browning, 2001).

While it seems natural to apply cluster analysis (Everitt et al., 2011) as a formal numerical approach to address DSM clustering, it turns out not directly suitable due to the presence of interactive components (i.e., the components that can be classified to multiple modules) (Eppinger and Browning, 2012). Therefore, some meta-heuristic approaches have been proposed for solving DSM clustering (e.g., genetic algorithms by Yu et al., (2007)). Yet, meta-heuristic approaches are relatively complex for product engineers as compared to cluster analysis. Thus, this paper applies and amends the methods from cluster analysis to address DSM clustering. There are two numerical strategies in the solution process. Firstly, the proposed method seeks for filtering the components that cause the “noise” in the clustering process. Then, the product’s modules can emerge more sharply in the results. Secondly, cluster analysis is utilized to preliminarily organize the input DSM, and that allows rooms for subsequent procedures to finalize the details of the solutions.

The rest of the paper is organized as follows. Section 2.2 provides the literature review. Section 2.3 explains the technical details of the DSM model and minimum description length

(MDL) that is used to evaluate the solution quality of DSM clustering. Section 2.4 presents the proposed method, with the details of matrix-based hierarchical clustering (MHC) and the three-phase methodical procedure. Section 2.5 demonstrates the proposed method via two literature examples and presents the comparison results. Section 2.6 concludes the paper.

2.2. Literature Review

In product architecture, clustering algorithms generally seeks for identification of highly coupled elements and grouping them into modules, leading to a set of modules that have less interaction with one another. This basic concept can be found in the work of Alexander (1964), and the general review of DSM clustering can be found in Browning (2001). The early clustering methods are usually heuristic such as the distance penalty method (Pimmler and Eppinger, 1994) due to its practicality. Further developments have involved some formal search methods such as simulated annealing (Fernandez, 1998) and genetic algorithm (Yu et al., 2007). To evaluate the quality of clustering results, the early work tends to minimize the module linkages (Alexander, 1964). In the context of product architecture, Sharman and Yassine (2004) examined the formation of modules towards the overall system benefit. Besides, Yu et al. (2007) proposed the concept of "description length" to evaluate clustering solutions via the comparison with the "ideal" situation.

In the application of DSM clustering, Shekar et al. (2011) implemented the DSM approach to manage the complexity in concurrent engineering design of aircraft. They demonstrated that using the DSM approach facilitates the flow of information and leads to significant reduction in development time as compared to sequential design of aircraft. Ko (2013) proposed the fuzzy DSM that combines fuzzy set theory and DSM to capture the dependency strength between product components. Accordingly, they used partitioning and tearing algorithms to restructure the DSM and identify modules of the product from the informational structure perspective. It is intended to optimize product architecture in terms of reducing cost and decreasing number of component iterations. Cheng et al. (2012) proposed a method based on DSM and axiomatic design to consider functional requirements and physical structure in product architecture. Rojas Arciniegas and Kim (2011) proposed a framework for identifying optimal module definition for a family of products by considering component sharing across all or some of the products. In problem solving, both Rojas Arciniegas and Kim (2011) and Cheng et al. (2012) used genetic algorithm (GA) to realize optimal product architecture.

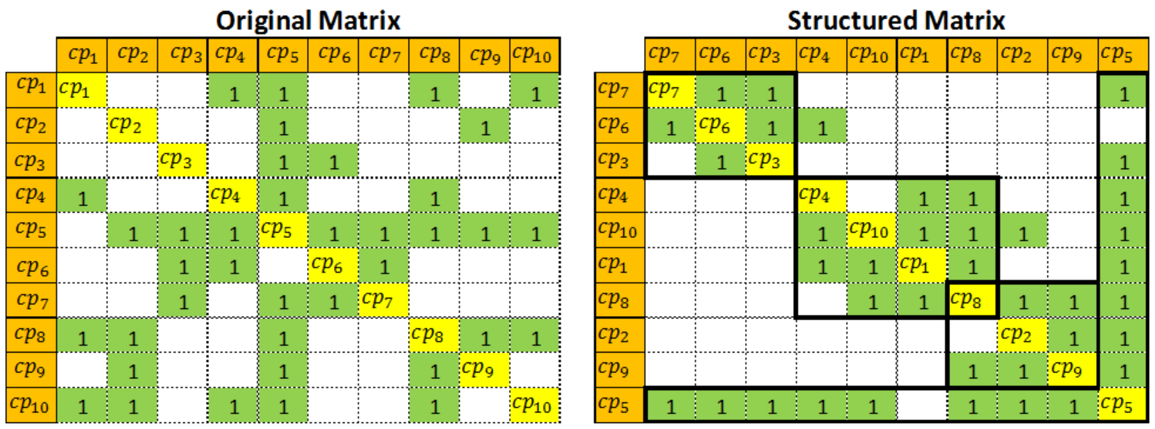
While cluster analysis is a well-established area (Everitt et al., 2011), the related solution techniques were not usually found in DSM clustering. One possible reason is that DSM clustering needs to suggest "overlapping modules", which some components are shared by few modules. This type of clustering solutions is not directly addressed by the conventional techniques in cluster analysis (e.g., hierarchical clustering or k-means clustering). Consequently, the application of genetic algorithm (GA), as reviewed before, has become a relatively popular approach. While GA is a powerful computational technique, it generally requires higher computing skills from users for desirable results such as the task of parameter tuning (Man et al., 1996, Khajavirad and Michalek, 2008). This motivates the research of using the techniques from cluster analysis for DSM clustering.

2.3. Product Architecture and Design Structure Matrix

2.3.1 Matrix representation

In the context of product architecture, Design Structure Matrix (DSM) has been used to capture the connections between product components (Eppinger and Browning, 2012). Let $CP = \{cp_1, cp_2, \ldots, cp_n\}$ be the set of n product components. Then, let $\Phi = [\phi_{ij}]$ be the product DSM, which is a $n \times n$ square matrix, and the matrix entry $[\phi_{ij}]$ captures the relational strength. In this context, a relation between two components is established if they receive and/or provide material, energy or information to another. Particularly, the matrix entry ϕ_{ij} is nonzero if the *i*th component receives something (i.e., material, energy or information) from the *j*th component. Otherwise, ϕ_{ij} is equal to zero. Alternately, the product DSM, $\Phi = [\phi_{ij}]$, can also be viewed as a non-symmetric, directional matrix.

In the context of product architecture, a product DSM can be "architected" via the identification of product modules and their interactions. This process is referred to as DSM clustering, which involves the permutations of the matrix's rows/columns and the definitions of modules. The goal of DSM clustering is to minimize the interactions between the formed modules (Yu et al., 2007). Figure 2-1 shows the example of an original matrix, which can be transformed to a structured matrix as the output of DSM clustering.



Original Matrix

	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	cp_8	cp_9	cp_{10}
cp_1	cp_1			1	1			1		1
cp_2		cp_2			1				1	
cp_3			cp_3		1	1				
cp_4	1			cp_4	1			1		
cp_5		1	1	1	cp_5	1	1	1	1	1
cp_6			1	1		cp_6	1			
cp_7			1		1	1	cp_7			
cp_8	1	1			1			cp_8	1	1
cp_9		1			1			1	cp_9	
cp_{10}	1	1		1	1			1		cp_{10}

Structured Matrix

	cp_7	cp_6	cp_3	cp_4	cp_{10}	cp_1	cp_8	cp_2	cp_9	cp_5
cp_7	cp_7	1	1							1
cp_6	1	cp_6	1	1						
cp_3		1	cp_3							1
cp_4				cp_4		1	1			1
cp_{10}				1	cp_{10}	1	1	1		1
cp_1				1	1	cp_1	1			1
cp_8					1	1	cp_8	1	1	1
cp_2								cp_2	1	1
cp_9							1	1	cp_9	1
cp_5	1	1	1	1	1		1	1	1	cp_5

Figure 2-1 DSM clustering: input and output

In product architecture, three types of components are classified in a structured matrix: module, interactive and bus components (Eppinger and Browning, 2012). Bus components are referred to the components that have relations with most of other components, and they are usually treated as the base or platform to coordinate with other modules. In Figure 2-1, cp_5 is the bus component. Module components are referred to the components that mainly have relations with components of the same modules and with the bus components. In Figure 2-1, $\{cp_3, cp_6, cp_7\}$ form a module, and they are module components. In contrast, interactive components have considerable relations with components of adjacent modules, and they represent the local interactions between two modules. For example, cp_8 is the interactive component in Figure 2-1 as it is "overlapped" with two adjacent modules.

Let $MD = \{md_1, md_2, \ldots, md_m\}$ be the set of m modules, and each module consists of subsets of components. Let cl_i be the number of components that belong to the ith module (i.e., cl_i is the size of the ith module). Also, let IC and BC be the sets of interactive and bus components, respectively. To illustrate, the structured matrix in Figure 2-1 shows that $md_1 = \{cp_7, cp_6, cp_3\}$, $md_2 = \{cp_4, cp_{10}, cp_1, cp_8\}$, $md_3 = \{cp_8, cp_2, cp_9\}$, $cp_8 \in IC$ and $cp_5 \in BC$.

In product architecture, the classification of components provides important information to design and manage product variants in design customization. To illustrate, Figure 2-2 shows the schematic view of product architecture for the DSM in Figure 2-1. This schematic shows that the module $\{cp_3, cp_6, cp_7\}$ as md_1 can be exchangeable if its interface with the bus component cp_5 can be maintained and standardized. By the formation of modules, the utility of DSM clustering is to highlight the important interfaces that should be closely controlled or standardized to effectively manage the variety of products (Sosa et al., 2003).

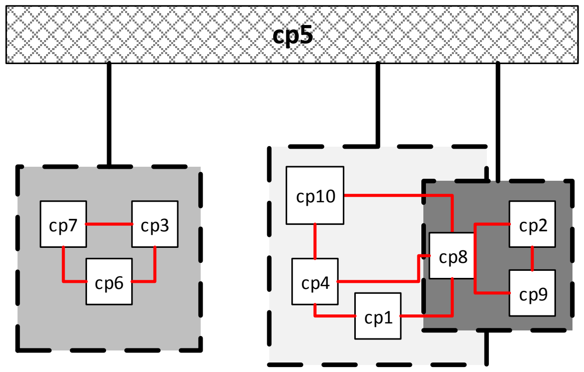


Figure 2-2 Schematic view of product architecture with a bus and one interactive component

2.3.2 Minimum description length

To evaluate the quality of the structured matrix after DSM clustering, this paper adopts the minimum description length (MDL) from Yu et al. (2007) as the objective measure for the comparison study. Conceptually, MDL envisions an ideal case that has no nonzero matrix entries (i.e., ϕ_{ij}) outside the module blocks and no zero matrix entries inside the module and bus blocks. Then, the "description length" is used to examine and quantify the difference between the proposed structured matrix and the ideal case.

Based on Yu et al. (2007), MDL consists of three measures: model description and two types of mismatches. The model description length, denoted as f_1, is used to examine the "data bits" required to describe a given number of modules, and it is formulated as follows.

$$f_1 = \sum_{i=1}^{m} \left(\log_2 n + cl_i \log_2 n \right) \tag{1}$$

Two types of mismatches, type I and II, are used to examine the <u>zero</u> matrix entries <u>inside</u> the module blocks and the <u>nonzero</u> matrix entries <u>outside</u> the module blocks, respectively. For brevity in the discussions, type I mismatches are labeled as "zero-inside", and type II mismatches are labeled as "nonzero-outside". Let S_1 be the number of "zero-inside" and S_2 be the number of "nonzero-outside". Then, the lengths of mismatches, f_2 for type I and f_3 for type II, are formulated as follows.

$$f_2 = S_1\left(2\log_2 n + 1\right) \tag{2}$$

$$f_3 = S_2\left(2\log_2 n + 1\right) \tag{3}$$

Accordingly, the total description length (denoted as f) can be formulated as a weighted sum function below with the weighting factors w_1, w_2 and w_3.

$$f = w_1 f_1 + w_2 f_2 + w_3 f_3 \tag{4}$$

As further explanation of MDL can be found in Yu et al. (2007), it should be noted that this original work has reported a procedure to tune the weighting factors to mimic the preferences of human experts. Yet, since the proposed method of this paper does not require the weighting factors for problem solving, the three measures (i.e., f_1, f_2 and f_3) are used for multi-objective comparisons in this study.

2.4. Methodology

This paper mainly uses hierarchical clustering as the core technique for DSM clustering, and its algorithmic overview will be provided in Section 2.4.1. Section 2.4.2 will develop the procedure that utilizes hierarchical clustering to address DSM clustering systematically.

2.4.1 Matrix-based hierarchical clustering (MHC)

In this paper, hierarchical clustering is based on the traditional agglomerative procedure (Everitt et al., 2011), which can be highlighted in two steps: similarity calculation and tree building. The similarity calculation is to quantify how appropriate two objects should be grouped in one cluster. Based on the similarity values, tree building is applied to group the objects progressively to render how similar objects should be clustered together.

From the traditional technique, matrix-based hierarchical clustering (MHC) is applied with two matrix-specific aspects. Firstly, the similarity calculation is based on the information presented in the original matrix. Secondly, beyond the formation of clusters, MHC also focuses on the sequences of the matrix's rows and columns. Specifically, it is intended to arrange the "similar" rows and columns close to each other so that the re-arranged matrix can show some kind of organized patterns. Procedurally, MHC is organized in two major steps: similarity analysis and sorting analysis, which will be elaborated in the following.

2.4.1.1 Similarity analysis procedure

In the context of DSM clustering, two components are similar if they have relations with some common components. For example, by checking the rows of cp_6 and cp_7 in Figure 2-1, they are said "similar" because they both have the "receive" relations with cp_3, cp_6 and cp_7 in common. The similarity value will be higher if there are more common components. The same idea can also be applied to the DSM's column to check for the "provide" relations.

By referencing the work of Li (2010), the min/max coefficient is adapted, and the similarity values between the ith and jth components based on rows (denoted as R_{row_ij}) and columns (denoted as R_{col_ij}) can be determined using the following equations.

$$R_{row\_ij} = \frac{\sum_k \min(\varphi_{ik}, \varphi_{jk})}{\sum_k \max(\varphi_{ik}, \varphi_{jk})} \qquad (5)$$

$$R_{col\_ij} = \frac{\sum_k \min(\varphi_{ki}, \varphi_{kj})}{\sum_k \max(\varphi_{ki}, \varphi_{kj})} \tag{6}$$

Row coupling matrix

	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	cp_8	cp_9	cp_{10}
cp_1	-	0.1	0.1	0.8	0.4	0.1	0.1	0.6	0.3	0.8
cp_2	0.1	-	0.2	0.2	0.3	0	0.2	0.5	0.8	0.3
cp_3	0.1	0.2	-	0.2	0.3	0.4	0.8	0.1	0.2	0.1
cp_4	0.8	0.2	0.2	-	0.3	0.1	0.1	0.4	0.3	0.7
cp_5	0.4	0.3	0.3	0.3	-	0.4	0.4	0.5	0.4	0.5
cp_6	0.1	0	0.4	0.1	0.4	-	0.6	0	0	0.1
cp_7	0.1	0.2	0.8	0.1	0.4	0.6	-	0.1	0.1	0.1
cp_8	0.6	0.5	0.1	0.4	0.5	0	0.1	-	0.7	0.7
cp_9	0.3	0.8	0.2	0.3	0.4	0	0.1	0.7	-	0.4
cp_{10}	0.8	0.3	0.1	0.7	0.5	0.1	0.1	0.7	0.4	-

Column coupling matrix

	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	cp_8	cp_9	cp_{10}
cp_1	-	0.3	0	0.5	0.4	0	0	0.7	0.1	0.6
cp_2	0.3	-	0.1	0.3	0.6	0.1	0.1	0.6	0.8	0.5
cp_3	0	0.1	-	0.3	0.3	1	0.8	0.1	0.1	0.1
cp_4	0.5	0.3	0.3	-	0.4	0.3	0.3	0.6	0.1	0.5
cp_5	0.4	0.6	0.3	0.4	-	0.3	0.2	0.7	0.4	0.4
cp_6	0	0.1	1	0.3	0.3	-	0.8	0.1	0.1	0.1
cp_7	0	0.1	0.8	0.3	0.2	0.8	-	0.1	0.2	0.2
cp_8	0.7	0.6	0.1	0.6	0.7	0.1	0.1	-	0.4	0.7
cp_9	0.1	0.8	0.1	0.1	0.4	0.1	0.2	0.4	-	0.3
cp_{10}	0.6	0.5	0.1	0.5	0.4	0.1	0.2	0.7	0.3	-

Figure 2-3 Similarity matrices

Notably, the row (column) similarity is referred to the "receive" ("provide") relations between two components. The min operation in the numerator is used to quantify the common components, and the max operation is used for normalization (so that the similarity value is between zero and one). Figure 2-3 shows the similarity matrices that record the similarity values between any two components for rows and columns. It should be noted that these similarity matrices are symmetric (e.g., $R_{row\_ij} = R_{row\_ji}$).

2.4.1.2 Sorting analysis

Given the similarity matrices as the input, the purpose of sorting analysis is to bring similar components close to each other in the original matrix. This involves the rearrangement of the matrix's rows and columns so that two similar components are placed next to each other in the sorted matrix. The basic idea of sorting analysis is to utilize tree building in hierarchical clustering. By building the tree (or dendrogram), the highly similar components tend to be clustered earlier in the agglomerative procedure. Then, the nodes of the resulting tree can be used to indicate the sequence for rearranging the matrix's rows and columns.

In DSM clustering, due to the asymmetric "receive" and "provide" relations, the similarity values between the *i*th and *j*th components according to row and column similarity matrices are not the same (i.e., $R_{row\_ij} \neq R_{col\_ij}$). To build the tree, a single similarity matrix is required to capture the similarity values among *n* components. In this study, it is considered that similarity values are not quite compensatory. That is, if the value of R_{row_ij} is high and R_{col_ij} is low, the "receive" similarity (i.e., R_{row_ij}) should not be reduced as reflected in the overall similarity. Thus, the max function is used to aggregate two similarity values, and the formulation of the overall similarity (denoted as R_{ij}) is provided as follows.

$$R_{ij} = \max\left(R_{row\_ij}, R_{col\_ij}\right) \tag{7}$$

Once the overall similarity of any two components is obtained, two algorithms can be carried for sorting analysis: tree construction and tree node sequencing. The details of these two algorithms can be found in Li (2010). For the readers' reference, Figure 2-4 provides the basic procedural steps of these algorithms. In brief, the tree construction algorithm takes the bottom-up approach (i.e., agglomerative) that groups the components based on their similarity values progressively. This represents the traditional procedure of constructing a dendrogram in hierarchical clustering. In contrast, the tree node sequencing algorithm is unique to MHC, and it focuses on switching the tree branches from the tree's top to bottom in order to bring similar tree nodes (or components) closer to each other.

Tree Construction Algorithm

Step 1: Pick two components entities that yield the highest similarity value.

Step 2: Label the leaves of the tree according to the picked components.

Step 3: Form the branch of the tree by combining the leaves (or branches). The vertical axis of the tree is labeled with the similarity values. The leaves (or branches) are merged to form a new branch at their similarity value.

Step 4: Update the similarity matrix through the average distance of the two picked components (say, i and j). The formulation is $R_{(ij)k} = (R_{ik}+R_{jk})/2$ where the subscript ij refers to the newly combined branch.

Step 5: Repeat Steps 1 to 4 until the similarity matrix cannot be further reduced.

Tree Node Sequencing Algorithm

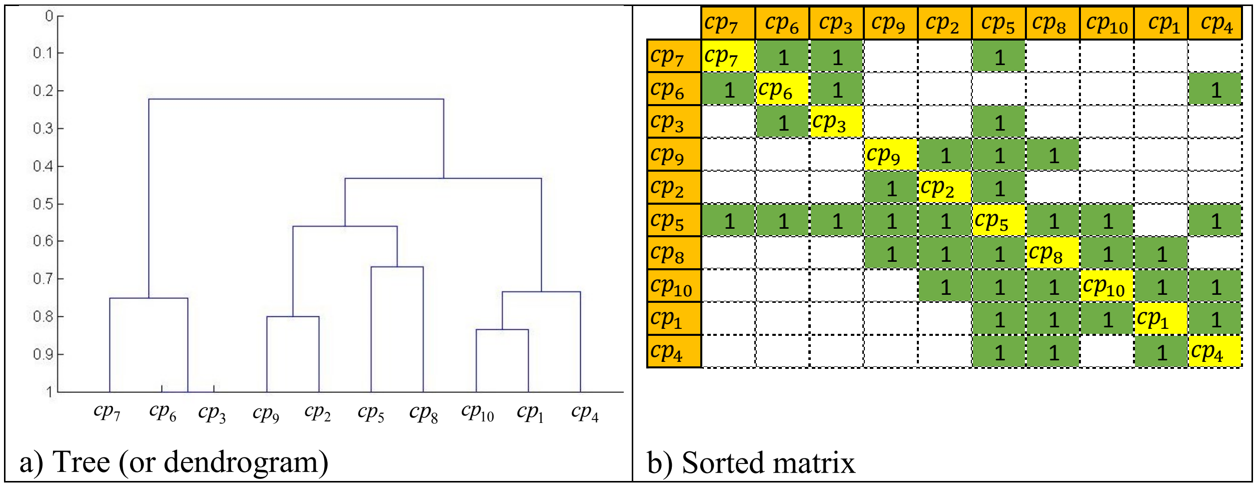
Step 1: Pick top two branches (say, branches x and y) considered for switching. Take the adjacent branch as the reference (say, branch z).

Step 2: if branch x has a higher similarity value with branch z and it is placed farther to branch z as compared to branch y, switch the positions of branches x and y.

Step 3: Proceed to the next top branches. Repeat Steps 1 and 2 until the bottom of the tree is reached.

Figure 2-4 Algorithms for tree construction and tree node sequencing

To demonstrate, the sample matrix in Figure 2-1 is used, and the resulting tree and sorted matrix are shown in Figure 2-5. Consider two components, cp_3 and cp_6, from the original matrix. The similarity values are $R_{row\_36} = 0.4$ and $R_{col\_36} = 1.0$ as shown in Figure 2-3. Based on Equation (8), the overall similarity value is $R_{36} = \max(0.4, 1.0) = 1.0$, which is the highest similarity value. In tree construction, cp_3 and cp_6 are then joined to form a branch as shown in Figure 2-5a. Given the same hierarchical tree structure, the tree node sequencing algorithm helps to determine the order of components that whether it should be [cp_3, cp_6] or [cp_6, cp_3]. Then, the node sequence of the tree is used to re-order the row and column sequences of the original matrix, as shown in Figure 2-5b.



	cp_7	cp_6	cp_3	cp_9	cp_2	cp_5	cp_8	cp_{10}	cp_1	cp_4
cp_7	cp_7	1	1			1				
cp_6	1	cp_6	1							1
cp_3		1	cp_3			1				
cp_9				cp_9	1	1	1			
cp_2				1	cp_2	1				
cp_5	1	1	1	1	1	cp_5	1	1		1
cp_8				1	1	1	cp_8	1	1	
cp_{10}					1	1	1	cp_{10}	1	1
cp_1						1	1	1	cp_1	1
cp_4						1	1		1	cp_4

Figure 2-5 Resulting matrix after sorting analysis

2.4.1.3 Remarks on MHC and DSM clustering

In this paper, MHC is treated as an algorithm package that takes the original matrix as the input and produces similarity matrices, tree and sorted matrix as the outputs. The further research investigation is how to utilize MHC for DSM clustering. As shown in Figure 2-5b, the sorted matrix after MHC shows some "approximate" structure that can be beneficial to DSM clustering. For example, it is observed that the component set $\{cp_7, cp_6, cp_3\}$ can be classified as one module. However, how to further classify the interactive components is not an obvious task. Then, the purpose of the next sub-section is to develop a procedure that utilizes MHC for yielding a structured matrix as the outcome of DSM clustering.

2.4.2 Procedure for DSM clustering

As discussed previously, the presence of interactive components has incurred the difficulty to directly apply cluster analysis for DSM clustering. Therefore, the methodical strategy is to identify bus and interactive components systematically so that the modules can emerge in a discernible pattern. In this paper, MHC is adapted for methodology development since it does not assume the number of clusters and supports some preliminary sorting of the matrix structure. The proposed methodology consists of three phases, which are described as follows.

- ❖ Phase 1: Component filtering. Some components (namely, bus and isolated components) do not have a clear membership to a single module, and they often introduce noises for the formation of modules. This phase is intended to identify and remove this kind of components for the subsequent clustering process.
- ❖ Phase 2: Approximate structure formation. After removing bus and isolated components, MHC is applied to obtain an approximate structure of DSM and determine the proper number of modules.
- ❖ Phase 3: Partitioning analysis. Given an approximate structure of DSM, partition points are systematically applied to define the boundary of modules, thus revealing the interactive and module components.

2.4.2.1 Phase 1: Component filtering

Two types of components want to be filtered, and they are bus and isolated component. Bus components serve like a platform that is connected to most of the components, and they can be identified quite readily from the original matrix. In contrast, isolated components are not connected to many components but they somehow have relations to multiple modules. Since bus and isolated components cannot be assigned to particular modules, their presence in the clustering process can somehow blur the modules' boundary. To address this issue, MHC is first applied to the original matrix as a filtering step to help identify bus and isolated components. Such a result has been illustrated in Figure 2-5b for the sample matrix.

In the definition, a component is identified as "bus" if it has relations with more than or equal to 70% of the total number of components (not including itself). Let $B = [b_{ij}]$ be the binary version of product DSM (i.e., $\Phi = [\phi_{ij}]$). That is, $b_{ij} = 1$ if ϕ_{ij} is nonzero; otherwise, $b_{ij} = 0$.

Let BC be the set of bus components. Recall that n is the total of components. Then, the members of BC (i.e., bus components) can be defined in Equation (8). To illustrate, consider the sorted matrix in Figure 2-5b, in which only cp_5 is identified as "bus", and both of its row and column have relations with 8/9 = 88.9% of total components (not including itself).

$$BC = \{cp_i \mid cp_i \in CP, \sum_j b_{ij} / (n-1) \geq 0.70 \text{ or } \sum_j b_{ji} / (n-1) \geq 0.70\} \quad (8)$$

Notably, a component is identified as "bus" even if it only has intensive relations in rows (i.e., $\Sigma_j b_{ij}$) or columns (i.e., $\Sigma_j b_{ji}$). For example, an information hub is still identified as "bus" if it receives most signals from other components (i.e., its row has many nonzero matrix entries) and provides only few outputs (i.e., its column has many zero matrix entries). Considerably, bus elements implicate a "platform" concept that can take intensive "provide" or "receive" relations. Thus, a bus element is defined based on this "either-or" condition.

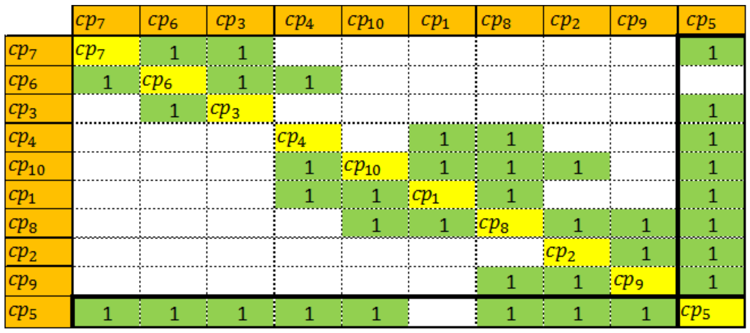
Regarding isolated component, it is identified as the component that does not have a relation with its adjacent component(s) in the sorted matrix. Notably, the sorted matrix obtained from MHC tends to give a diagonal matrix with nonzero entries packed along the diagonal. Thus, it is not common to have isolated components (e.g., there are no isolated components in Figure 2-5b). Yet, when they emerge, they show having relations with multiple modules that are not adjacent to them in the sorted matrix. As a result, the presence of isolated components can blur the modules' boundary. After identifying the bus and isolated components, they are removed from the original matrix, leading to a "filtered" matrix that is ready for the next phase.

2.4.2.2 Phase 2: Approximate Structure Formation

Given the matrix without the bus and isolated components, MHC is performed again. The result will be the sorted version of the “filtered matrix”. To form the “approximate structure”, the sorted filtered matrix is placed first. Then, the isolated and bus components (identified in the previous phase) will be placed on at the right and bottom sides of the matrix. Figure 2-6 shows the approximate structure of the sample matrix.

The approximate structure is treated as the intermediate result before finalizing the module and interactive components. One of its advantages is that we do not need to assume the number of modules to obtain this approximate structure. In contrast, the “black-box” type of approaches often requires the number of modules as the algorithmic input at the beginning (Yu et al., 2007). Also, as the approximate structure has already organized the “similar” components, it may already present some meaningful information for domain experts (e.g., $\{cp_3, cp_6, cp_7\}$ as one module). Potentially, domain experts can even be able to finalize the structured matrix as the result of DSM clustering.

To complete the methodical procedure to address DSM clustering algorithmically, the next phase, partitioning analysis, is developed to systematically define the module and interactive components based on the approximate structure.



	cp_7	cp_6	cp_3	cp_4	cp_{10}	cp_1	cp_8	cp_2	cp_9	cp_5
cp_7	cp_7	1	1							1
cp_6	1	cp_6	1	1						
cp_3		1	cp_3							1
cp_4				cp_4		1	1			1
cp_{10}				1	cp_{10}	1	1	1		1
cp_1				1	1	cp_1	1			1
cp_8					1	1	cp_8	1	1	1
cp_2								cp_2	1	1
cp_9							1	1	cp_9	1
cp_5	1	1	1	1	1		1	1	1	cp_5

Figure 2-6 Approximate structure of the sample matrix

2.4.2.3 Phase 3: Partitioning Analysis

The purpose of partitioning analysis is to define the module and interactive components using partition points, and the concept of partition points can be found in Li (2011). In brief, a partition point is used to define the boundary of modules in a diagonal matrix. Figure 2-7 shows a schematic of a partition point placed in a diagonal matrix, where the dark spots represent the nonzero matrix entries. Each partition point defines a vertical and a horizontal line, and modules are referred to the diagonal blocks. Then, the positions of partition points can specify the final structured matrix. To specify a point's position, a coordinate system is used, where the coordinate (0,0) is referred to the top left corner.

To place partition points properly, two guidelines are used. Firstly, since DSM clustering in this paper is based on a square matrix, the partition points are only placed along the diagonal. Secondly, a good partition point tends to separate the components that share low similarity values. Otherwise, if two components have a high similarity value, they should be grouped in the same module. To support the placement of partition points, the tree and similarity values from MHC in Phase 2 will be used.

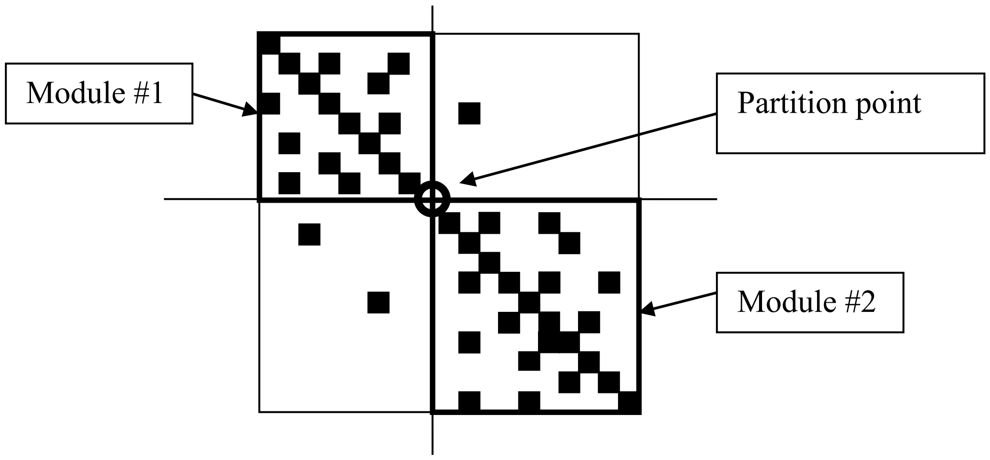


Figure 2-7 Concept of partition point in a diagonal matrix

Firstly, the proper number of modules is determined through tree cutting. Figure 2-8 shows the tree from Phase 2 (i.e., without bus and isolated components). A threshold is set to cut the tree at the similarity value of 0.6. Each branch that is crossed by this cut line and has multiple components is considered as one module. As indicated by the ovals in Figure 2-8, the sample matrix has three modules. To define three modules, we need to place a minimum of two partition points.

Secondly, the quality of partition points is assessed based on the similarity values. For example, consider Figure 2-9a, where a partition point (3,3) is placed that directly separates cp_3 and cp_4. Then, the similarity value of cp_3 and cp_4 is used to examine the quality of this partition point. Along the diagonal for the possible partition points, Table 2-1 shows their corresponding similarity values. As discussed before, a good partition point should have a low similarity value. In the case of the sample matrix, two partition points are needed, and thus (3,3) and (7,7) are chosen due to their lowest similarity values to preliminarily define three modules, as shown in Figure 2-9a.

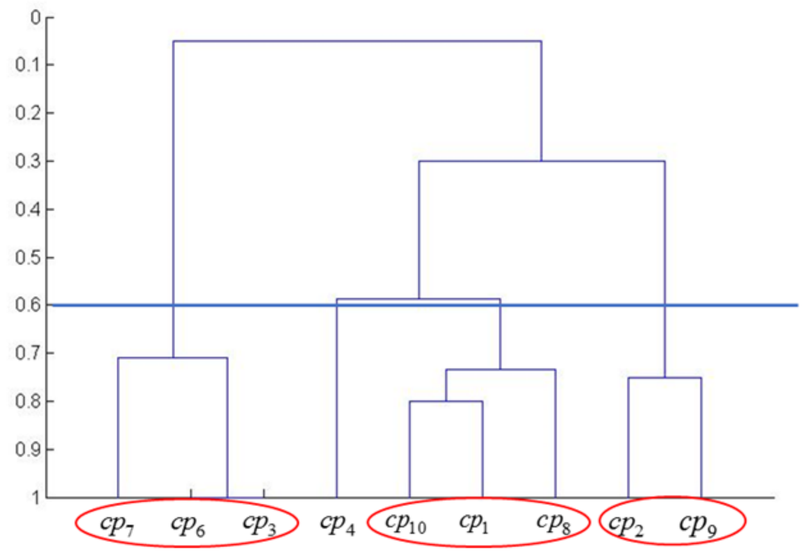


Figure 2-8 Tree cutting and number of modules

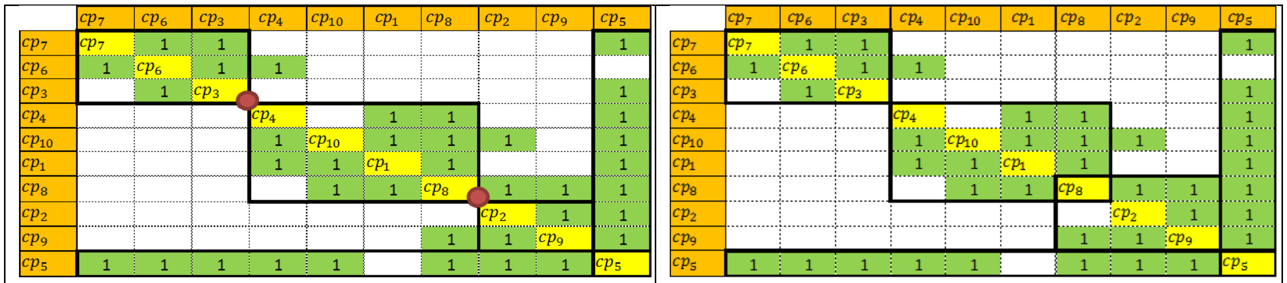
Table 2-1 Partition points and similarity values

Partition point	Similarity value
(1,1)	0.66
(2,2)	1.00
(3,3)	0.17
(4,4)	0.60
(5,5)	0.80
(6,6)	0.80
(7,7)	0.50
(8,8)	0.75

Once the preliminary modules are defined, the next step is to identify interactive components. Notably, each module can be confined by a start point at the top left and the end point at the bottom right. For example, the second module in Figure 2-9a has the start point (3,3) and the end point (7,7). Interactive components are introduced if the start point moves to the left upward or the end point to the right downward, as illustrated in Figure 2-9a. Then, the partitioning strategy is to examine whether the start and end points of each preliminary module should be moved to form interactive components. Provided below is the procedure to move start/end points and identify interactive components.

- ❖ Step 1: check the start point of a module. If the coordinate of the start point is (x, x), check the tentative module with the start point (x-1, x-1). This tentative module has added some new matrix entries. If new nonzero entries are more than new zero entries, accept the new start point and check the next one (x-2, x-2). Otherwise, stop the checking.
- ❖ Step 2: check the end point of a module. If the coordinate of the end point is (y, y), check the tentative module with the end point (y+1, y+1). Again, if new nonzero entries are more than new zero entries, accept the new end point and check the next one (y+2, y+2). Otherwise, stop the checking.
- ❖ Step 3: move to the next module and repeat Steps 1 and 2.

To illustrate, consider the end point of the second preliminary module in Figure 2-9a (i.e., (7,7)). Since it is an end point, the next check will be (8,8), which introduces more new zero entries (total: 6) than new nonzero entries (total: 2). Thus, the end point should stay at (7,7). In contrast, consider the start point of the third preliminary module (i.e., (7,7)). Since it is a start point, the next check will be (6,6). As the number of new nonzero entries (total: 3) is more than the number of new zero entries (total: 1), the new end point (6,6) is accepted, as illustrated in Figure 2-9b. Table 2-2 also provides the start and end points of the sample matrix in Figure 2-9 to highlight the function of partitioning analysis.



	cp_7	cp_6	cp_3	cp_4	cp_{10}	cp_1	cp_8	cp_2	cp_9	cp_5
cp_7	cp_7	1	1							1
cp_6	1	cp_6	1	1						
cp_3		1	cp_3							1
cp_4				cp_4		1	1			1
cp_{10}				1	cp_{10}	1	1	1		1
cp_1				1	1	cp_1	1			1
cp_8					1	1	cp_8	1	1	1
cp_2								cp_2	1	1
cp_9							1	1	cp_9	1
cp_5	1	1	1	1	1		1	1	1	cp_5

a) Formation of preliminary modules

	cp_7	cp_6	cp_3	cp_4	cp_{10}	cp_1	cp_8	cp_2	cp_9	cp_5
cp_7	cp_7	1	1							1
cp_6	1	cp_6	1	1						
cp_3		1	cp_3							1
cp_4				cp_4		1	1			1
cp_{10}				1	cp_{10}	1	1	1		1
cp_1				1	1	cp_1	1			1
cp_8					1	1	cp_8	1	1	1
cp_2								cp_2	1	1
cp_9							1	1	cp_9	1
cp_5	1	1	1	1	1		1	1	1	cp_5

b) Final structured matrix

Figure 2-9 Preliminary modules and final structured matrix

Table 2-2 Start and end points of preliminary modules and final structure matrix

	Preliminary modules		Final structure matrix	
	Start point	End point	Start point	End point
Module 1	(0,0)	(3,3)	(0,0)	(3,3)
Module 2	(3,3)	(7,7)	(3,3)	(7,7)
Module 3	(7,7)	(9,9)	(6,6)	(9,9)

2.5. Application and Comparison Study

In this section, two application examples from the literature are used to examine the proposed method for DSM clustering. Particularly, the minimum description length (MDL) described in Section 2.3.2 is used to compare the literature results and investigate the features of the cluster analysis approach.

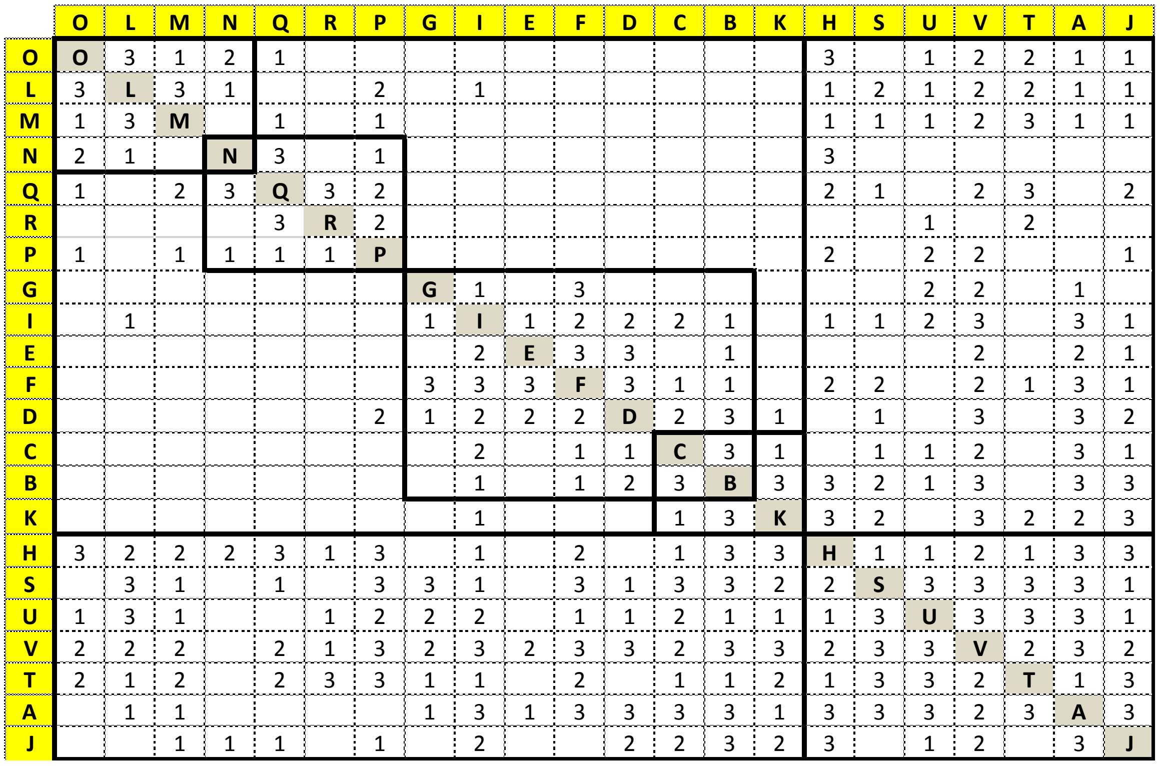
2.5.1 GM powertrain (GMPT)

The DSM example here is referred to the capture of the communication pattern among 22 teams in General Motors (GM) developing a powertrain system. Particularly, each matrix entry indicates the direction and intensity of communications between two teams. This DSM model was initially found in McCord and Eppinger (1993) and has been approached by the genetic algorithms (GA) for clustering (Yu et al., 2007). While the interactions here are referred to teams (i.e., not components), the matrix contents are equivalent to the models in DSM clustering. Due to the availability of the literature results, this DSM is used in this paper for comparison purpose.

By applying the proposed method to this GMPT matrix, seven "bus" teams are first identified. Yet, no "isolated" teams are found. By proceeding to Phases 2 and 3 of the proposed method, the proper number of modules is determined as four, and thus three partition points are initially required to define four preliminary modules. The result of this approximate structure is provided in Figure 2-10. To finish the procedure in partitioning analysis, the start and end points of these preliminary modules are checked, and some of them are moved to form interactive components. The final structured matrix is obtained and shown in Figure 2-11.

	O	L	M	N	Q	R	P	G	I	E	F	D	C	B	K	H	S	U	V	T	A	J
O	**O**	3	1	2	1											3		1	2	2	1	1
L	3	**L**	3	1			2		1							1	2	1	2	2	1	1
M	1	3	**M**		1		1									1	1	1	2	3	1	1
N	2	1		**N**	3		1									3						
Q	1		2	3	**Q**	3	2									2	1		2	3		2
R					3	**R**	2											1		2		
P	1		1	1	1	1	**P**									2		2	2			1
G								**G**	1		3							2	2		1	
I		1						1	**I**	1	2	2	2	1		1	1	2	3		3	1
E									2	**E**	3	3		1					2		2	1
F								3	3	3	**F**	3	1	1		2	2		2	1	3	1
D							2	1	2	2	2	**D**	2	3	1		1		3		3	2
C									2		1	1	**C**	3	1		1	1	2		3	1
B									1		1	2	3	**B**	3	3	2	1	3		3	3
K									1				1	3	**K**	3	2		3	2	2	3
H	3	2	2	2	3	1	3		1		2		1	3	3	**H**	1	1	2	1	3	3
S		3	1		1		3	3	1		3	1	3	3	2	2	**S**	3	3	3	3	1
U	1	3	1			1	2	2	2		1	1	2	1	1	1	3	**U**	3	3	3	1
V	2	2	2		2	1	3	2	3	2	3	3	2	3	3	2	3	3	**V**	2	3	2
T	2	1	2		2	3	3	1	1		2		1	1	2	1	3	3	2	**T**	1	3
A		1	1					1	3	1	3	3	3	3	1	3	3	3	2	3	**A**	3
J			1	1	1		1		2			2	2	3	2	3		1	2		3	**J**

Figure 2-10 Approximate structure (GMPT)



	O	L	M	N	Q	R	P	G	I	E	F	D	C	B	K	H	S	U	V	T	A	J
O	O	3	1	2	1											3		1	2	2	1	1
L	3	L	3	1			2		1							1	2	1	2	2	1	1
M	1	3	M		1		1									1	1	1	2	3	1	1
N	2	1		N	3		1									3						
Q	1		2	3	Q	3	2									2	1		2	3		2
R					3	R	2											1		2		
P	1		1	1	1	1	P									2		2	2			1
G								G	1		3							2	2		1	
I		1						1	I	1	2	2	2	1		1	1	2	3		3	1
E									2	E	3	3		1					2		2	1
F								3	3	3	F	3	1	1		2	2		2	1	3	1
D							2	1	2	2	2	D	2	3	1		1		3		3	2
C									2		1	1	C	3	1		1	1	2		3	1
B									1		1	2	3	B	3	3	2	1	3		3	3
K									1				1	3	K	3	2		3	2	2	3
H	3	2	2	2	3	1	3		1		2		1	3	3	H	1	1	2	1	3	3
S		3	1		1		3	3	1		3	1	3	3	2	2	S	3	3	3	3	1
U	1	3	1			1	2	2	2		1	1	2	1	1	1	3	U	3	3	3	1
V	2	2	2		2	1	3	2	3	2	3	3	2	3	3	2	3	3	V	2	3	2
T	2	1	2		2	3	3	1	1		2		1	1	2	1	3	3	2	T	1	3
A		1	1					1	3	1	3	3	3	3	1	3	3	3	2	3	A	3
J			1	1	1		1		2			2	2	3	2	3		1	2		3	J

Figure 2-11 Final structured matrix (GMPT)

To conduct the comparison study, the results based on manual clustering (McCord and Eppinger, 1993) and genetic algorithm (GA) (Yu et al., 2007) are used, and their matrix solutions can be found in Appendix. Table 2-3 shows the numerical results in view of minimum description length (MDL), and the identified bus and interactive components. Notably, instead of reporting the aggregated MDL, this study provides the values of f_1, f_2 and f_3 (as formulated in Equations (1), (2) and (3)) without assuming the values of weights. It is intended to provide a more comprehensive view in the comparison.

Table 2-3 Comparison results of the GMPT example

	MDL			Bus Team	Interactive Team
	f_1	f_2	f_3		
Proposed method	133.78	1563.80	52.86	A, H, J, S, T, U, V	B, C, N
GA	133.78	1183.65	138.86	H, S, T, U, V	A, D, I
Manual	160.54	1467.99	92.58	H, S, T, U, V	A, B, D, I, K, O

Compared with the literature results (both GA and manual), one key difference of the solution by the proposed method is the inclusion of A and J in the bus. By checking these two teams, they do have communications with many other teams, and thus it is reasonable to include them in the bus. Notably, these new bus teams impact the value of f_2 since they introduce more zero matrix entries inside the modules and bus (i.e., more "zero-inside"). Yet, the value of f_3 can be improved due to less nonzero matrix entries outside the modules and bus (i.e., less "nonzero-outside").

As observed, there is a trade-off between the values of f_2 and f_3 (e.g., a good value of f_2 is traded with a poor value of f_3). Technically, none of these DSM solutions is dominated in view of Pareto optimality, and the solution by the proposed method is not poorer than other solutions. Arguably, it is considered that f_3 is more important in view of the solution quality since the "nonzero outside" often implicates additional efforts in terms of design and communication (e.g., specific meeting for discussing the interface information). Yet, the "zero inside" does not often incur additional efforts. This view supports the advantage of the solution by the proposed method.

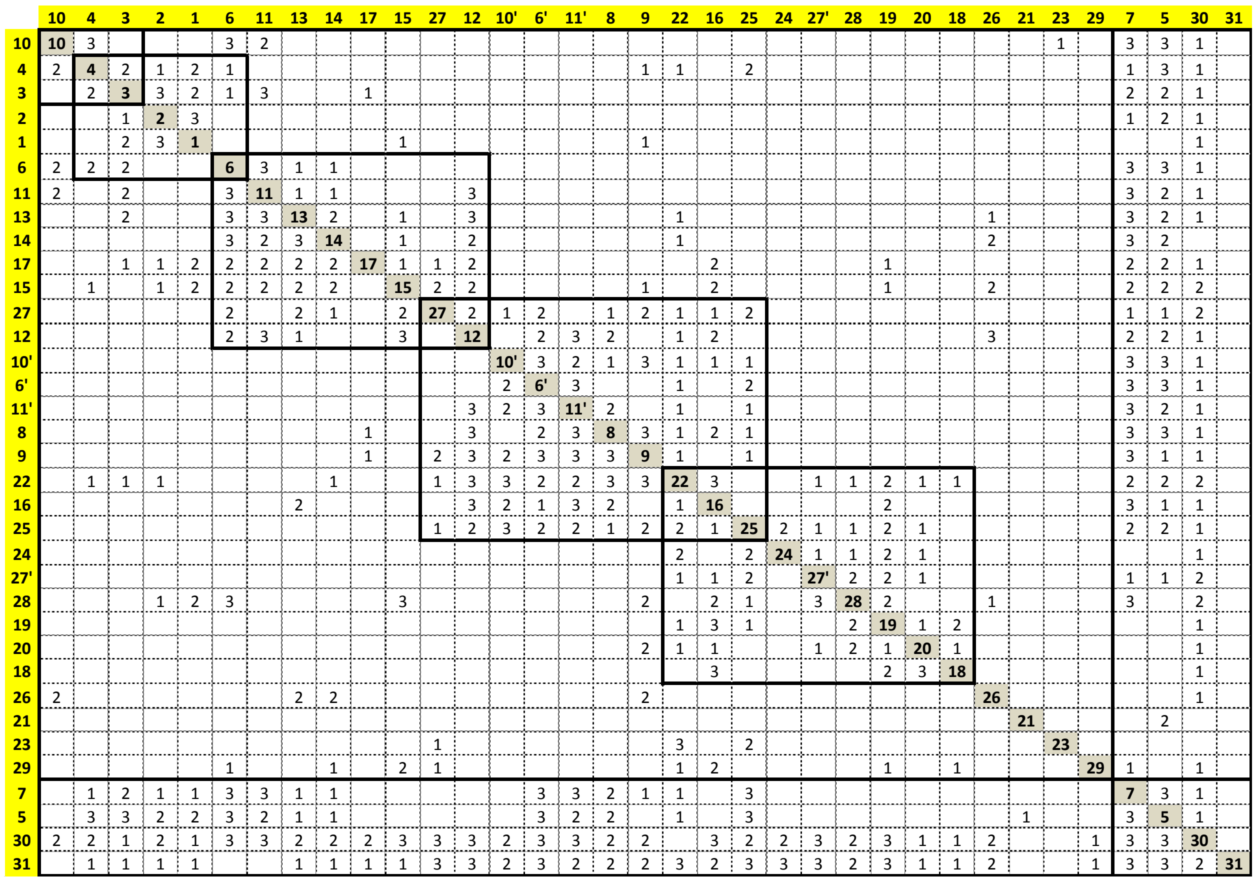
2.5.2 Gas turbine generator (GAS)

The DSM example here is about the interactions of sub-systems of a gas turbine, and sub-systems are equivalent to the component concept in this paper. The initial DSM model can be found in Sharman and Yassine (2004), and the corresponding GA solution can be found in Yu et al. (2007). In Phase 1 of the proposed method, MHC is first applied to the original matrix, and the result is shown in Figure 2-12. In this result, it is found that sub-systems #21, #26, #29 and #23, located at both ends, have no relations with their adjacent sub-systems. Thus, they are classified as isolated sub-systems. Along with four bus sub-systems (i.e., #5, #7, #30, #31), these isolated sub-systems are arranged at the side of the matrix. The matrix with the remaining sub-systems is used for Phases 2 and 3. Figure 2-13 shows the final structured matrix obtained by the proposed method.

	21	26	4	1	2	3	10	6	11	13	14	17	15	12	7	5	10'	6'	11'	8	9	25	22	31	30	27	27'	28	19	16	18	20	24	29	23
21	21															2																			
26		26					2			2	2										2				1										
4			4	2	1	2	2	1							1	3					1	2	1		1										
1				1	3	2							1								1				1										
2				3	2	1									1	2									1										
3			2	2	3	3		1	3			1			2	2									1										
10			3				10	3	2						3	3									1										1
6			2			2	2	6	3	1	1				3	3									1										
11						2	2	3	11	1	1			3	3	2									1										
13		1				2		3	3	13	2		1	3	3	2							1		1										
14		2						3	2	3	14		1	2	3	2							1												
17				2	1	1		2	2	2	2	17	1	2	2	2									1	1			1	2					
15		2	1	2	1			2	2	2	2		15	2	2	2					1				2	2			1	2					
12		3						2	3	1			3	12	2	2		2	3	2			1		1					2					
7			1	1	1	2		3	3	1	1				7	3		3	3	2	1	3	1		1										
5	1		3	2	2	3		3	2	1	1				3	5		3	2	2		3	1		1										
10'															3	3	10'	3	2	1	3	1	1		1					1					
6'															3	3	2	6'	3			2	1		1										
11'														3	3	2	2	3	11'	2		1	1		1										
8												1		3	3	3		2	3	8	3	1	1		1					2					
9												1		3	3	1	2	3	3	3	9	1	1		1	2									
25														2	2	2	3	2	2	1	2	25	2		1	1	1	1	2	1		1	2		
22			1		1	1					1			3	2	2	3	2	2	3	3		22		2	1	1	1	2	3	1	1			
31		2	1	1	1	1				1	1	1	1	3	3	3	2	3	2	2	2	3	3	31	2	3	3	2	3	2	1	1	3	1	
30		2	2	1	2	1	2	3	3	2	2	2	3	3	3	3	2	3	3	2	2	2			30	3	3	2	3	3	1	1	2	1	
27								2		2	1		2	2	1	1	1	2		1	2	2	1		2	27				1					
27'															1	1						2	1		2		27'	2	2	1		1			
28		1		2	1			3					3		3						2	1			2		3	28	2	2					
19																						1	1		1			2	19	3	2	1			
16										2				3	3	1	2	1	3	2			1		1				2	16					
18																									1				2	3	18	3			
20																					2		1		1		1	2	1	1	1	20			
24																						2	2		1		1	1	2			1	24		
29								1			1		2		1								1		1	1			1	2	1			29	
23																						2	3			1									23

Figure 2-12 Sorted matrix after applying MHC for the first time (GAS)

Table 2-4 shows the numerical results from the proposed method as compared to the solutions by manual clustering and genetic algorithm (GA) (Yu et al., 2007). The manual and GA matrix solutions can be found in Appendix. Compared with the solution by GA, one difference of the solution by the proposed method is the inclusion of sub-system #31in the bus. By checking the matrix in detail, it is found that this sub-system #31 is like a sink that has "receive" relations with many other sub-systems (as shown in its matrix row) but basically has no "provide" relations (as shown in its matrix column). This explains why our solution has a poorer f_2 value (i.e., more "zero inside") but a better f_3 value (i.e., less "nonzero outside") as compared to the GA solution.



	10	4	3	2	1	6	11	13	14	17	15	27	12	10'	6'	11'	8	9	22	16	25	24	27'	28	19	20	18	26	21	23	29	7	5	30	31
10	**10**	3				3	2																							1		3	3	1	
4	2	**4**	2	1	2	1												1	1		2											1	3	1	
3		2	**3**	3	2	1	3			1																						2	2	1	
2			1	**2**	3																											1	2	1	
1			2	3	**1**						1							1																1	
6	2	2	2			**6**	3	1	1																							3	3	1	
11	2		2			3	**11**	1	1				3																			3	2	1	
13			2			3	3	**13**	2		1		3						1									1				3	2	1	
14						3	2	3	**14**		1		2						1									2				3	2		
17			1	1	2	2	2	2	2	**17**	1	1	2							2					1							2	2	1	
15		1		1	2	2	2	2	2		**15**	2	2					1		2					1			2				2	2	2	
27						2		2	1		2	**27**	2	1	2		1	2	1	1	2											1	1	2	
12						2	3	1			3		**12**		2	3	2		1	2								3				2	2	1	
10'														**10'**	3	2	1	3	1	1	1											3	3	1	
6'														2	**6'**	3			1		2											3	3	1	
11'													3	2	3	**11'**	2		1		1											3	2	1	
8										1			3		2	3	**8**	3	1	2	1											3	3	1	
9										1		2	3	2	3	3	3	**9**	1		1											3	1	1	
22		1	1	1					1			1	3	3	2	2	3	3	**22**	3			1	1	2	1	1					2	2	2	
16								2					3	2	1	3	2		1	**16**					2							3	1	1	
25												1	2	3	2	2	1	2	2	1	**25**	2	1	1	2	1						2	2	1	
24																			2		2	**24**	1	1	2	1								1	
27'																			1	1	2		**27'**	2	2	1						1	1	2	
28				1	2	3					3							2		2	1		3	**28**	2			1				3		2	
19																			1	3	1			2	**19**	1	2							1	
20																		2	1	1			1	2	1	**20**	1							1	
18																				3					2	3	**18**							1	
26	2							2	2									2										**26**						1	
21																													**21**				2		
23												1							3		2									**23**					
29						1			1		2	1							1	2					1		1				**29**	1		1	
7		1	2	1	1	3	3	1	1						3	3	2	1	1		3											**7**	3	1	
5		3	3	2	2	3	2	1	1						3	2	2		1		3								1			3	**5**	1	
30	2	2	1	2	1	3	3	2	2	2	3	3	3	2	3	3	2	2		3	2	2	3	2	3	1	1	2			1	3	3	**30**	
31		1	1	1	1			1	1	1	1	3	3	2	3	2	2	2	3	2	3	3	3	2	3	1	1	2			1	3	3	2	**31**

Figure 2-13 Final structured matrix (GAS)

In general, the sub-system that shows many nonzero entries just in a row or column is still classified as "bus" in this paper. It is because this condition is aligned with the "platform" concept related to "bus". However, as observed in this example, it is not necessary the case in the GA approach. Notably, the GA approach needs to have a single fitness function (i.e., an aggregated MDL) in the optimization process, which does not purposely distinguish the nature of sub-systems with many nonzero entries in either rows or columns. Thus, the GA approach can classify sub-system #31 as "interactive" as long as the better f_2 value can compensate the weaker f_3 value. This highlights the nature of solution strategy by GA versus cluster analysis.

Table 2-4 Comparison results of the GAS example

	MDL			Bus sub-system	Interactive sub-system
	f_1	f_2	f_3		
Proposed method	225.68	2799.6	356.44	#5, #7, #30, #31	#3, #4, #6, #12, #16, #22, #25, #27,
GA	230.81	2630.8	360.27	#5, #7, #30	#3, #6, #11, #12, #16, #22, #25, #27, #31
Manual	271.85	3854.1	198.93	#17, #22, #25, #27, #28, #30, #31	#3, #5, #6, #7, #11

2.5.3 Comparison with GA as a methodical approach

Computationally, the difference between GA and the proposed method is that GA is developed based on optimization formalism. The optimality of the solutions from GA depends on the strategy to explore the solution space (e.g., chromosome definition and mutation rate). The proposed method is based on cluster analysis, which consists of finite loops of calculations (e.g., similarity calculations and tree construction). Thus, the computational effort of the proposed method is more manageable than that of GA.

The trade-off of the computational effort is the optimality of the solutions that GA has a track record to yield nearly optimal solutions (Gen and Cheng, 2000). In DSM clustering, the objective MDL essentially has three sub-objectives (i.e., f_1, f_2 and f_3). Notably, the GA solutions tend to give the highest value of f_3, indicating that they do not specifically care "nonzero outside". Comparatively, the manual solutions tend to yield lower f_3 values, imply that the domain experts may concern module interactions/interfaces (as "nonzero outside") more than the "zero inside". Without conclusively judging which solution is better, it is at least fair to note that setting the weights for f_1, f_2 and f_3 for automatic optimization processes is not trivial in this case. Yet, the weights of f_1, f_2 and f_3 need to be specified for the algorithmic execution in GA.

Compared with GA, the proposed method for DSM clustering allows more rooms for the inputs from domain experts in problem solving. First of all, it does not assume an aggregated objective for the algorithmic process. Instead, it depends on the formulation of “similarity” that focuses on the reasoning why two elements should be grouped together. It is considered that the similarity equations (i.e., (5) and (6)) of this paper are simple, and domain experts can even do their own formulations based on context-specific information. Also, since bus elements are filtered at the early stage, it is actually viable for domain experts to identify bus elements by themselves with their own judgments and criteria.

Furthermore, the algorithmic support of the proposed method lies in the result of the sorted matrix, showing an “approximate structure” that cannot be easily obtained by visual inspection or manual efforts. In turn, as the sorted matrix has preliminarily grouped the “similar” elements, it provides an opportunity for domain experts to do further re-arrangement and form modules and interactive elements on their own. For example, domain experts can set the number of modules after inspecting the sorted matrix.

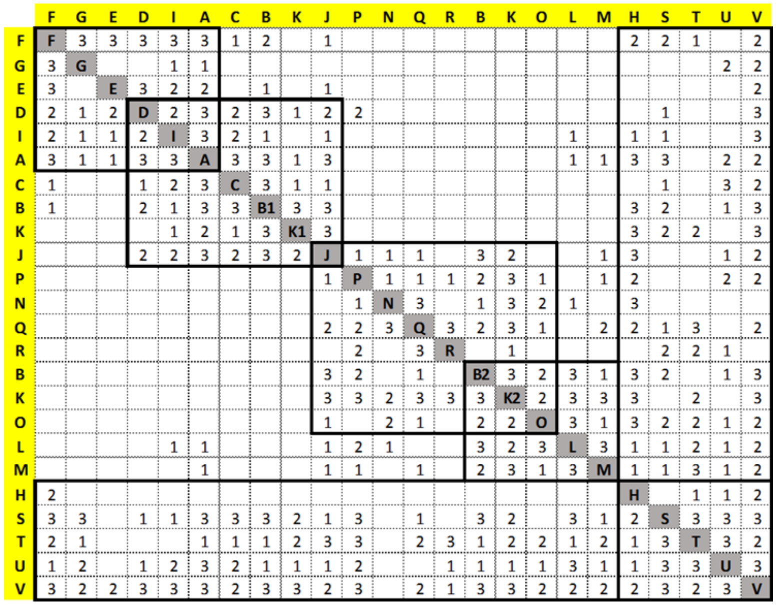
In between the manual and GA approaches, it is considered that the proposed method based on cluster analysis has provided a “middle route”. On the one hand, the proposed method has applied algorithmic procedures to organize some complex details automatically (e.g., approximate structure formation via tree construction). On the other hand, the proposed method allows the participation from domain experts to execute their own judgement such as identification of bus elements and formation of modules from the sorted matrix. Considerably, the proposed method can provide some means for domain experts to develop the final solutions based on some contextual information that is not explicitly articulated in the model.

2.6. Conclusions

To support the development of product architectures, this paper employs DSM to represent and visualize the relations among components of the product. Then, the specific problem is DSM clustering that seeks for the identification of modular, interactive and bus components. The methodical approach is based on matrix-based hierarchical clustering (MHC), and it consists of three phases: (1) components filtering, (2) approximate structure formation, and (3) partitioning analysis. The proposed method has been demonstrated via two DSM examples from literature, and the results are compared in view of three sub-objectives of MDL. While the proposed method can yield comparable results in view of existing solutions by manual and GA approaches, the method's advantage lies in its flexibility that allows domain experts to provide their judgment in the midst of the solution process (e.g., define modules over the approximate structure). It is considered that the proposed method demonstrates that cluster analysis can be treated as an alternate numerical strategy (in addition to optimization) to solve DSM clustering problems.

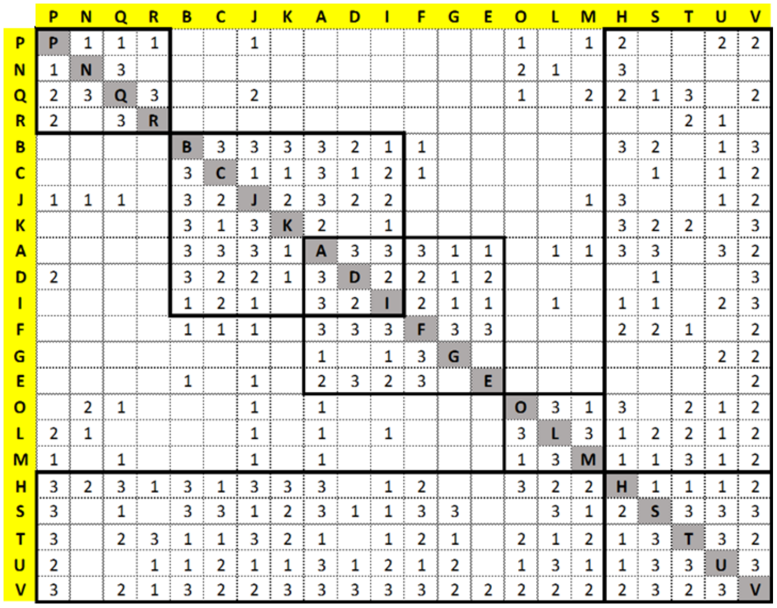
2.7. Appendix

To conduct comparison study, the results based on manual clustering and genetic algorithm (GA) for GMPT example are presented in Figure 2-14 and Figure 2-15 respectively.



	F	G	E	D	I	A	C	B	K	J	P	N	Q	R	B	K	O	L	M	H	S	T	U	V
F	F	3	3	3	3	3	1	2		1										2	2	1		2
G	3	G			1	1																	2	2
E	3		E	3	2	2		1		1														2
D	2	1	2	D	2	3	2	3	1	2	2										1			3
I	2	1	1	2	I	3	2	1		1								1		1	1			3
A	3	1	1	3	3	A	3	3	1	3								1	1	3	3		2	2
C	1			1	2	3	C	3	1	1											1		3	2
B	1			2	1	3	3	B1	3	3										3	2		1	3
K					1	2	1	3	K1	3										3	2	2		3
J				2	2	3	2	3	2	J	1	1	1		3	2			1	3			1	2
P										1	P	1	1	1	2	3	1		1	2			2	2
N											1	N	3		1	3	2	1		3				
Q										2	2	3	Q	3	2	3	1		2	2	1	3		2
R											2		3	R		1					2	2	1	
B										3	2		1		B2	3	2	3	1	3	2		1	3
K										3	3	2	3	3	3	K2	2	3	3	3		2		3
O										1		2	1		2	2	O	3	1	3	2	2	1	2
L					1	1				1	2	1			3	2	3	L	3	1	1	2	1	2
M						1				1	1		1		2	3	1	3	M	1	1	3	1	2
H	2																			H		1	1	2
S	3	3		1	1	3	3	3	2	1	3		1		3	2		3	1	2	S	3	3	3
T	2	1				1	1	1	2	3	3		2	3	1	2	2	1	2	1	3	T	3	2
U	1	2		1	2	3	2	1	1	1	2			1	1	1	1	3	1	1	3	3	U	3
V	3	2	2	3	3	3	2	3	3	2	3		2	1	3	3	2	2	2	2	3	2	3	V

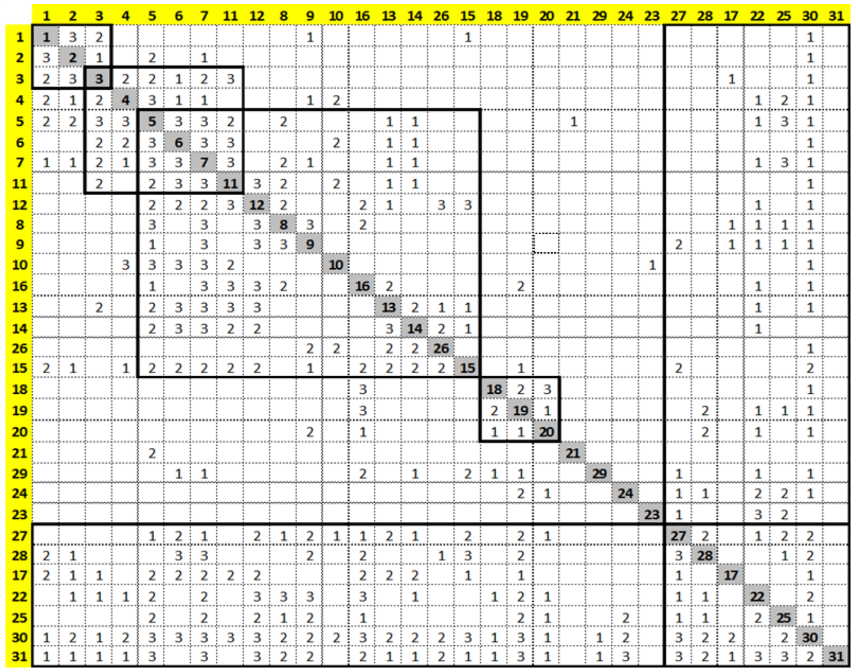
Figure 2-14 Manually clustered GMPT



	P	N	Q	R	B	C	J	K	A	D	I	F	G	E	O	L	M	H	S	T	U	V
P	P	1	1	1			1								1		1	2			2	2
N	1	N	3												2	1		3				
Q	2	3	Q	3			2								1		2	2	1	3		2
R	2		3	R																2	1	
B					B	3	3	3	3	2	1	1						3	2		1	3
C					3	C	1	1	3	1	2	1							1		1	2
J	1	1	1		3	2	J	2	3	2	2						1	3			1	2
K					3	1	3	K	2		1							3	2	2		3
A					3	3	3	1	A	3	3	3	1	1		1	1	3	3		3	2
D	2				3	2	2	1	3	D	2	2	1	2					1			3
I					1	2	1		3	2	I	2	1	1		1		1	1		2	3
F					1	1	1		3	3	3	F	3	3				2	2	1		2
G									1		1	3	G								2	2
E					1		1		2	3	2	3		E								2
O		2	1				1		1						O	3	1	3		2	1	2
L	2	1					1		1		1				3	L	3	1	2	2	1	2
M	1		1				1		1						1	3	M	1	1	3	1	2
H	3	2	3	1	3	1	3	3	3		1	2			3	2	2	H	1	1	1	2
S	3		1		3	3	1	2	3	1	1	3	3			3	1	2	S	3	3	3
T	3		2	3	1	1	3	2	1		1	2	1		2	1	2	1	3	T	3	2
U	2			1	1	2	1	1	3	1	2	1	2		1	3	1	1	3	3	U	3
V	3		2	1	3	2	2	3	3	3	3	3	2	2	2	2	2	2	3	2	3	V

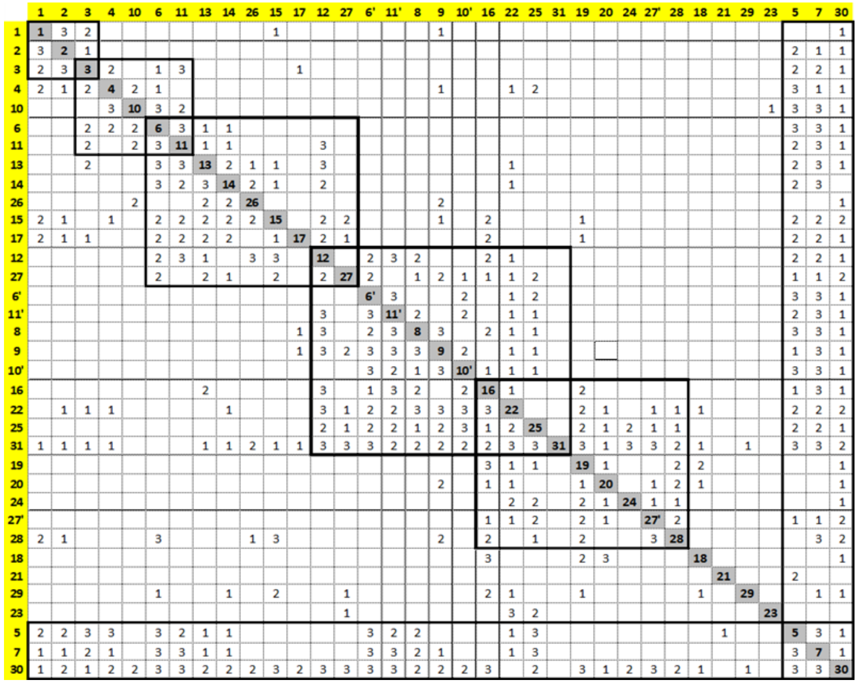
Figure 2-15 Clustering arrangement by GA for GMPT

Also the result based on manual clustering for GAS DSM is provided in Figure 2-16. The clustering arrangement by GA for GAS DSM can be found in Figure 2-17.



	1	2	3	4	5	6	7	11	12	8	9	10	16	13	14	26	15	18	19	20	21	29	24	23	27	28	17	22	25	30	31
1	1	3	2								1						1													1	
2	3	2	1		2		1																							1	
3	2	3	3	2	2	1	2	3																			1			1	
4	2	1	2	4	3	1	1				1	2																1	2	1	
5	2	2	3	3	5	3	3	2		2				1	1						1							1	3	1	
6			2	2	3	6	3	3				2		1	1															1	
7	1	1	2	1	3	3	7	3		2	1			1	1													1	3	1	
11			2		2	3	3	11	3	2		2		1	1															1	
12					2	2	2	3	12	2			2	1		3	3											1		1	
8					3		3		3	8	3		2														1	1	1	1	
9					1		3		3	3	9														2		1	1	1	1	
10				3	3	3	3	2				10												1						1	
16					1		3	3	3	2			16	2					2									1		1	
13			2		2	3	3	3	3					13	2	1	1											1		1	
14					2	3	3	2	2					3	14	2	1											1			
26											2	2		2	2	26														1	
15	2	1		1	2	2	2	2	2		1		2	2	2	2	15		1						2					2	
18													3					18	2	3										1	
19													3					2	19	1						2		1	1	1	
20											2		1					1	1	20						2		1		1	
21					2																21										
29						1	1						2		1		2	1	1			29			1			1		1	
24																			2	1			24		1	1		2	2	1	
23																								23	1			3	2		
27					1	2	1		2	1	2	1	1	2	1		2		2	1					27	2		1	2	2	
28	2	1				3	3				2		2			1	3		2						3	28			1	2	
17	2	1	1		2	2	2	2	2				2	2	2		1		1						1		17			1	
22		1	1	1	2		2		3	3	3		3		1			1	2	1					1	1		22		2	
25					2		2		2	1	2		1						2	1			2		1	1		2	25	1	
30	1	2	1	2	3	3	3	3	3	2	2	2	3	2	2	2	3	1	3	1		1	2		3	2	2		2	30	
31	1	1	1	1	3		3		3	2	2		2	1	1	2	1	1	3	1		1	3		3	2	1	3	3	2	31

Figure 2-16 Manually clustered GAS



	1	2	3	4	10	6	11	13	14	26	15	17	12	27	6'	11'	8	9	10'	16	22	25	31	19	20	24	27'	28	18	21	29	23	5	7	30
1	1	3	2								1							1																	1
2	3	2	1																														2	1	1
3	2	3	3	2		1	3					1																					2	2	1
4	2	1	2	4	2	1												1			1	2											3	1	1
10				3	10	3	2																									1	3	3	1
6			2	2	2	6	3	1	1																								3	3	1
11			2		2	3	11	1	1				3																				2	3	1
13			2			3	3	13	2	1	1		3								1												2	3	1
14						3	2	3	14	2	1		2								1												2	3	
26					2			2	2	26								2																	1
15	2	1		1		2	2	2	2	2	15		2	2				1		2				1									2	2	2
17	2	1	1			2	2	2	2		1	17	2	1						2				1									2	2	1
12						2	3	1		3	3		12		2	3	2			2	1												2	2	1
27						2		2	1		2		2	27	2		1	2	1	1	1	2											1	1	2
6'															6'	3			2		1	2											3	3	1
11'													3		3	11'	2		2		1	1											2	3	1
8												1	3		2	3	8	3		2	1	1											3	3	1
9												1	3	2	3	3	3	9	2		1	1											1	3	1
10'															3	2	1	3	10'	1	1	1											3	3	1
16								2					3		1	3	2		2	16	1			2									1	3	1
22		1	1	1					1				3	1	2	2	3	3	3	3	22			2	1		1	1	1				2	2	2
25													2	1	2	2	1	2	3	1	2	25		2	1	2	1	1					2	2	1
31	1	1	1	1				1	1	2	1	1	3	3	3	2	2	2	2	2	3	3	31	3	1	3	3	2	1		1		3	3	2
19																				3	1	1		19	1			2	2						1
20																		2		1	1			1	20		1	2	1						1
24																					2	2		2	1	24	1	1							1
27'																				1	1	2		2	1		27'	2					1	1	2
28	2	1				3				1	3							2		2		1		2			3	28						3	2
18																				3				2	3				18						1
21																														21			2		
29						1			1		2			1						2	1			1					1		29			1	1
23														1							3	2										23			
5	2	2	3	3		3	2	1	1						3	2	2				1	3								1			5	3	1
7	1	1	2	1		3	3	1	1						3	3	2	1			1	3											3	7	1
30	1	2	1	2	2	3	3	2	2	2	3	2	3	3	3	3	2	2	2	3		2		3	1	2	3	2	1		1		3	3	30

Figure 2-17 Clustering arrangement by GA for GAS

2.8. Author's Notes and Significance of Paper to Thesis

This chapter is devoted to identify modular structure of product which is a prerequisite for managing product variety. Modularity refers to product architecture design where product variants can be created by substituting different module combinations. In this way, this chapter identifies different sets of components in the product in a way to support product variety.

Knowing composition of modules and type of components in terms of module, bus, or interactive component plays a key role in other stages of product life cycle. For example, in terms of supply chain network design, modules of the products and their usage in final product along with demand for each product variants determine the structure of supply chain network. Structuring supply chain network based on modular structure of the product is discussed in the next Chapter.

Also in terms of manufacturing, information about modular structure of the product helps to identify product platforms in order to reduce costs. For instance, bus components act as platforms that other components mounted on them. Therefore, by mass producing these shared components, some production costs in terms of setup costs can be saved. Platform-based manufacturing for a family of products is discussed in Chapter 4.

In summary, this chapter is related with other chapters of thesis as modular structure of product that is identified in this chapter will be used in other chapters for managing product variety in other phases of product lifecycle (e.g. supply chain and manufacturing).

3. Hierarchical Clustering for Structuring Supply Chain Network in Case of Product Variety

ABSTRACT

To compete in the market, manufacturers often need to offer multiple product variants to different customers. Given such a challenge of product variety to supply chain, the postponement strategy has been adapted that the differentiating modules are handled at the later stage of the process to minimize the impacts from demand variations. Given the information of product variants and their mix ratios, this paper focuses on the problem of structuring supply chain network that indicates the precedence orders of suppliers and sub-assemblers. The solution approach is based on hierarchical clustering, in which the tree structure is applied to construct the supply chain network. One core technique is to investigate the coupling values between the modules by characterizing the grouping condition in the structuring process. A complexity measure is also adopted to compare the quality of resulting supply chain networks. Five numerical examples are utilized to demonstrate the applicability and effectiveness of the proposed clustering method.

3.1. Introduction

Offering product variety to satisfy a broad range of customers has been the essence of today's manufacturing. For example, BMW claims that "every vehicle that rolls of the belt is unique" and the number of possible variations in BMW 7 series could reach 10^{17} (Hu et al., 2008). Therefore, it is a challenging task to design a production system and supply chain to handle such high variety while maintaining quality and productivity at the same time. In order to cope with the challenges caused by product variety, manufacturers implement modular design (Wang et al., 2010), in which a product is decomposed into different functional modules, and the variety is achieved by offering several options for each module. A wide range of product variants can then be synthesized by combinational assembly of different modules.

Powered by modular product architecture, nowadays manufacturing activities become more spatially dispersed. As a result, manufacturers do not have to perform all manufacturing steps by themselves in a single facility location. Instead, some of the manufacturing and assembly processes can be assigned to suppliers, and the final assembler receives few sub-assemblies for further processing and producing the finished products (Zhang et al., 2013). Examples can be found in various industries like automobile and personal computers where customers are offered a variety of product options within a short delivery time. For instance, Volvo S80 model has 17 pre-assembled units that 11 of which are operated by suppliers (Wang et al., 2010). Assigning some of the assembly tasks to the suppliers allows manufacturers to exploit comparative advantages of different locations. However, it can dramatically increase the complexity of a supply chain system (Zhang et al., 2013).

In order to tackle challenges due to high product variety in supply chain, manufacturers implement postponement strategy. Postponement strategy which is enabled by modular product design is one key for mass customization (Feitzinger and Lee, 1997), and it can be used to increase supply chain responsiveness (Fisher, 1997). However, implementing postponement in a pre-defined supply chain network where the position and product of the suppliers and sub-assemblers are already fixed will limit its effectiveness. Therefore, unlike the supply chain configuration problems that often assume the fixed supply chain network and concern with the optimal selection of suppliers subject to various considerations like lead time and cost (Graves and Willems, 2005), this paper aims to construct the supply chain network based on the product variety information.

In this paper, a methodical procedure is developed to construct the supply chain network in a way to support postponement and reduce the complexity of the network. The structuring problem specifically considers two pieces of product variety information: (1) the number of different types of product variants to be offered, and (2) the demands of various product variants. Generally, if the number of variant types is small and most demands come from only few variants, the final assembler may choose to assemble most of the modules by themselves due to the economy of scale. Alternatively, if the number of variant types is high and most demands of different variants are generally dispersed, the final assembler may try to determine the modules that are required by most variants and assign them to the upstream suppliers. This approach is aligned with the concept of postponement that the final assembler can focus on product differentiation at later stages of supply chain network.

Considering this structuring problem, it has been observed that there is a similarity in the process of structuring supply chain network and hierarchical clustering as both try to bundle the elements in the multi-stage structuring process. Thus, this paper proposes hierarchical clustering as the numerical tool to facilitate the structuring process of supply chain network. The specific new concept of this paper is to formulate the coupling values based on the product variety information and construct the supply chain network accordingly.

The research results are relevant for supply chain managers who need to make decisions in different supply chain practices such as postponement, procurement policy and supplier selection. It offers managerial insights to enhance supply chain performance through considering postponement strategy in the structuring of supply chain network. Five sample cases are developed to demonstrate the applications of the proposed method on the decisions that should be made by supply chain managers in terms of postponement strategy as well as verifying the capability of the method to suggest a supply chain network with lower complexity.

The remaining part of this paper is organized as follows. Section 3.2 provides some background information and literature review on structuring supply chain network and its computational complexity. Section 3.3 describes the research problem along with the complexity model for a supply chain network. Section 3.4 establishes the methodical procedure for structuring supply chain network based on product variety information. Section 3.5 presents some numerical examples to show the application of the proposed method in supply chain practices. Finally, section 3.6 provides some closing remarks and describes our future research direction.

3.2. Background and Related Works

3.2.1. Supply chain network & postponement

The performance of a supply chain network for a product family is determined by design decisions of products, assembly processes and supply chain. Supply chains are often modeled as multi-stage assemblies and inventory networks (Huang et al., 2005). For example, Graves and

Willems (2005) developed a supply chain configuration optimization model that minimizes the total supply chain costs. However, they address the supply chain configuration problem in terms of selecting suppliers, processes and transportation modes for a supply chain network that has already been fixed. Similarly, Wong et al. (2011) evaluated postponement as an option to improve the performance of the supply chain system for the soluble coffee. Their results show that significant cost saving can be achieved by considering postponement in supply chain network.

According to the literature of supply chain network, improvements have been suggested based on the current (or fixed) structure of the supply chain network. The supply chain networks are usually developed based on the knowledge and expertise of the experts (like the notebook supply chain in Graves and Willems (2005)), and so far limited systematic methods have been proposed in literature about the structural construction of supply chain network. The most relevant work on structuring supply chain network can be found in Wang et al. (2010). In this work, they employ the concepts of information entropy and module-based product family architecture to structure assembly supply chain in order to minimize the complexity of the structure.

The structure of supply chain network is one key determinant of efficiency and complexity in a supply chain system. Nowadays, one important trend in supply chain management enabled by modular product design is the emergence of modular supply chain (Wang et al., 2009, Fine et al., 2005). The concepts of modular supply chain and non-modular supply chain are illustrated in Figure 3-1. In modular supply chain, final product apportions to sub-assemblies that are mostly outsourced to suppliers and therefore only few assembled modules will be delivered to the final assembler for the final assembly operation. Modular supply chain has many applications in automotive and aerospace industry (Hu et al., 2008). In contrast, non-modular supply chain mainly represents the case of mass production where the economy of scale is more important. Also, the complexity of the whole network is usually lower in this case as fewer elements (e.g., suppliers and sub-assemblers) are engaged in the process.

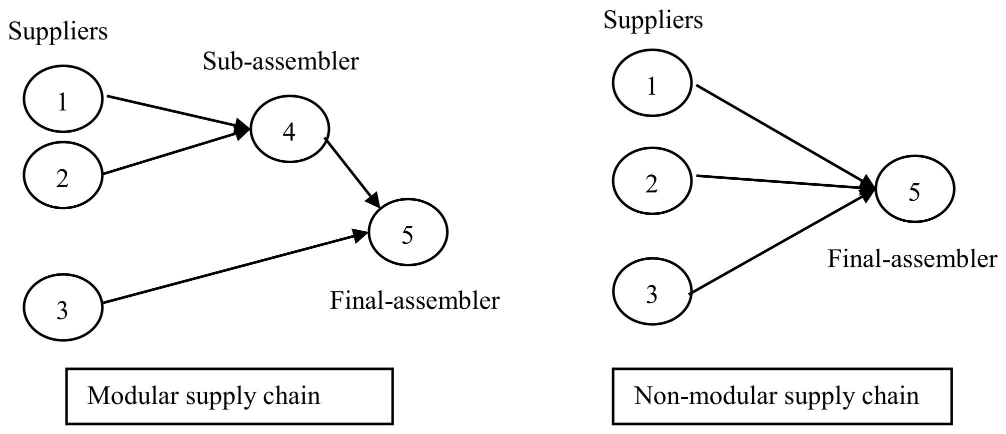


Figure 3-1 Modular supply chain Vs Non-modular supply chain

The concept of postponement has a long history in terms of both practical applications and academic literature. While the practical application of the concept can be traced back to the 1920s, the early empirical descriptions of postponement appeared in 1960s (Pagh and Cooper, 1998). In terms of literature, the postponement concept was established by Bucklin (1965). Postponement is also known as “delayed product differentiation” among manufacturing researchers since it concerns with delaying the product differentiation point in multi-product lines. The basic logic behind postponement is that the risk and uncertainty are tied with differentiating parts of the product. Therefore, to the extent that manufacturing and operation of these parts can be postponed, the risk, uncertainty, forecasting errors and complexity of manufacturing can be reduced (Kim, 2014). The development of the postponement concept has its root in increased demands for customized products. Modularity and high commonality of the modules are important characteristics of postponement. For instance, modular product architecture lets manufacturers provide the variety of end products on the basis of few generic platforms (Yang and Yang, 2010).

Postponement is a powerful strategy to achieve cost effective mass customization primarily through reducing inventory requirement. Specially, it requires more attention when a company offers different variants of a product. Postponement enables dramatic inventory cost reduction as there is less randomness in demand for base modules (e.g., platform) than demand for differentiating modules of product. It is because in many cases demands for different variants of the product are negatively correlated. Therefore, when the demand for one product variant is high, it is more likely that demand for some other product variants will be low (Ulrich and Eppinger, 2012).

In this research, a specific type of supply chain network is considered that each node can have one output, and the outputs of upstream nodes are funneled through downstream integrators ending up with one final node (final assembler) that produces the final product by putting together all the sub-assemblies (Wang et al.). Figure 3-2 shows a general form of supply chain network, in which node sp_i stands for the ith supplier ($i \in \{1, 2, \ldots, s\}$), and node sa_j represents the jth sub-assembler ($j \in \{1, 2, \ldots, r\}$). Also, there are two special nodes at both ends of supply chain network (*source* and *FA*). The node "*source*" can be viewed as a provider of raw materials and the node "*FA*" is the final assembler who yields final product variants. Except these two special nodes, other nodes have only one output edge for producing a set of module options or sub-assemblies. Besides, modular product design is considered that a product is apportioned to different modules, and several variants from each module can be offered to achieve product variety. Arguments for modular product design is usually based on economical concerns such as cost and time to market (Fine et al., 2005).

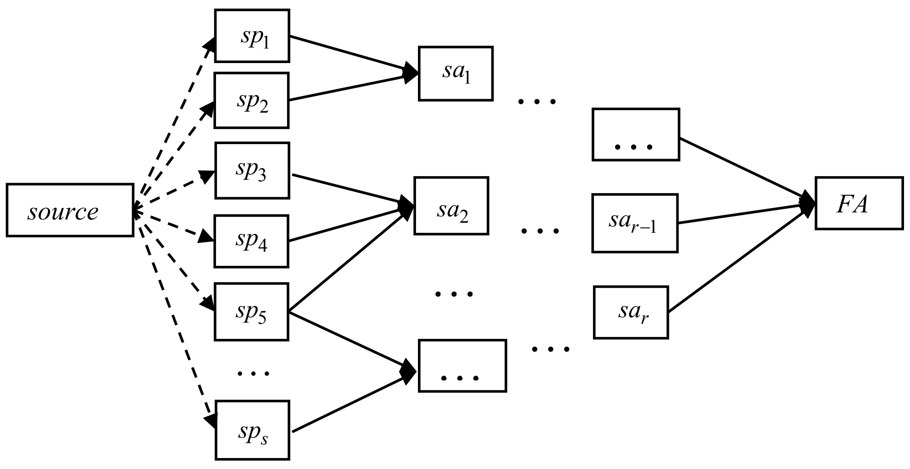


Figure 3-2 General supply chain network

3.2.2. Computational complexity of supply chain network

The problem of structuring supply chain network is to determine the supply chain network that minimizes the complexity measure. The challenge of such a structuring problem in manufacturing has been discussed in the review paper by Hu et al. (2011). The fundamental challenge is that the possible number of structures is numerous even for a small number of elements (e.g., modules or suppliers). For instance, the number of possible configurations for supply chain networks with n elements roughly lies between 2^{n-1} and $n!$ (Webbink and Hu, 2005). The early relevant works are found in the context of assembly sequencing, including the cut-set method (Baldwin et al., 1991) and algorithms based on relational model (Homem De Mello and Sanderson, 1991). Webbink and Hu (2005) employed the grouping corollary to generate possible system configurations analytically. Wang et al. (2011) developed a multi-objective optimization approach to balance the level of product variety and manufacturing complexity for a mix-model assembly system. They used the utopian point method along with genetic algorithms to solve this combinatorial problem. Zhu et al. (2012) used an integer programming formulation to find the optimal assembly sequence to minimize the complexity induced by product variety in mixed-model assembly line. Zhang et al. (2013) developed a bi-objective supply chain model to identify the structure of supply chain. They used branch and cut algorithm to solve the model and illustrated the benefits of non-modular supply chain structure in terms of cost reduction. Ko et al. (2013) worked on assembly decomposition decisions and developed an assembly decomposition model to improve product quality. Basically, the assembly decomposition problem is to divide the assembly into sub-assemblies that can be outsourced to the suppliers. These sub-assemblies are produced individually by suppliers and assembled together in the final assembly process (Ko et al., 2013).

In summary, in order to find the structure of supply chain network with minimum complexity, the basic approach is mainly based on three general steps: (1) generate all possible structures, (2) compute the complexity measures, and (3) select the structure that has minimum complexity (Hu et al., 2008). In this way, the first step of the process is the most challenging part due to computational complexity incurred by numerous possible configurations even for a small network. In terms of literature summary, the approaches to tackle such structuring problem are either based on some techniques to expedite the solution process (Koren and Shpitalni, 2010, Wang et al., 2010, Baldwin et al., 1991) or optimization techniques (Wang et al., 2011, Zhu et al., 2012, Wang et al., 2013, Ko et al., 2013). Compared with the structuring efforts in assembly sequence and supply chain, the application of hierarchical clustering as the solution approach is relatively limited in the literature. This paper is intended to contribute in this research direction and propose a method based on hierarchical clustering to structure supply chain network.

Notably, the structuring of supply chain network may be confused with the problem of supply chain configuration which is concerned with the optimal selection of the candidate suppliers subject to various considerations such as lead-time and cost (Graves and Willems, 2005). The problems of supply chain configuration assume the “fixed” supply chain network, termed as “generic supply chain network” (Huang et al., 2005) that indicated the precedence of parts to be supplied. Considerable literature is available on connecting the supply chain network with the product information such as product types (functional or innovative) (Fisher, 1997), product modularity (Salvador et al., 2002) and product architecture (Nepal et al., 2012).

3.3. Problem Statement and Complexity

3.3.1. Problem description

In general, supply chain can be considered as a network of nodes. These nodes represent the enterprises engaged in activities such as the supply of raw materials, assembly and delivery. More specifically, from a single manufacturer point of view, these nodes can represent the units that perform the tasks such as procurement of raw materials, fabrication of parts, assembly of components and delivery of finished products to target customers (Huang et al., 2005). In this way, the network of these nodes constitutes the structure of a supply chain network, and multiple options are often available to deploy such structural arrangements.

The challenge therefore is how to construct a supply chain network based on the available information to form an effective and efficient supply chain. This challenge is further complicated when a firm offers multiple variants of a product. Different product variants often share similarities in terms of modules and assembly processes despite distinctive features. This will affect the structure of supply chain network and can be a potential source of inventory cost reduction through applying postponement strategy.

For managing product variety, one basic piece of information is the set of product variants that are offered to the market. These product variants are different from each other in view of unique product features desired by some specific customers. Based on the work of Wang et al. (2010), the model is described as follows. Let $PV = \{pv_1, pv_2, \ldots, pv_n\}$ be the set of n product variants. To achieve product variety, all product variants share the same modular structure. Let $MD = \{md_1, md_2, \ldots, md_p\}$ be the set of p modules. Each module can offer more than one option and let $md_{i,j}$ represent the jth option of the ith module. Figure 3-3 shows three different variants of a product that consists of three modules and each module has two options. In this way, different product variants can be synthesized by selecting different options of each module, and a large number of product variants can be potentially obtained (e.g., there can be eight possible product variants in Figure 3-3).

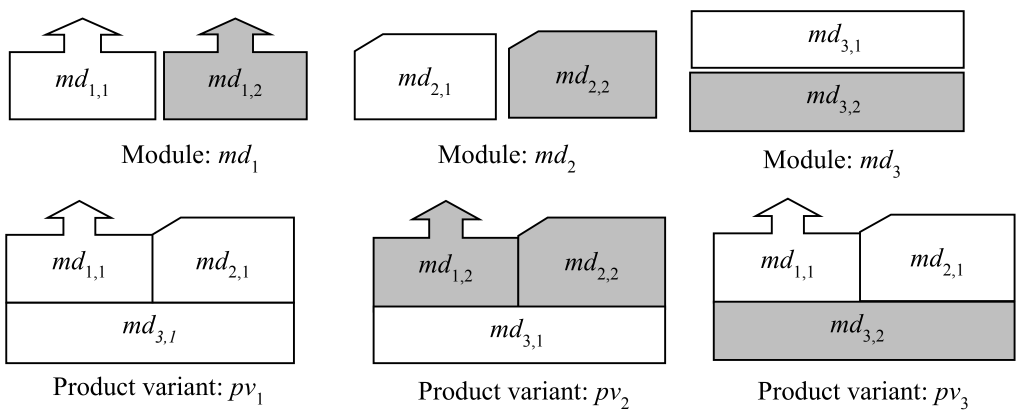


Figure 3-3 Illustration of modules and product variants

In addition, the mix ratio information is used to capture the fractional demands of product variants (Wang et al., 2010). Let q_i be the mix ratio of pv_i and $\sum q_i = 1$. Figure 3-4 shows the mix ratio of product variants presented in Figure 3-3. For instance, Figure 3-4 shows that pv_1 has a mix ratio of 0.7, indicating that pv_1 takes 70% of the total demand of all product variants.

At this point, the research problem is to determine the structure of supply chain network given the information of *PV*, *MD* and the mix ratios. In practice, the information of *MD* should be determined by the product design team, and the information of *PV* and mix ratios should come from the marketing team.

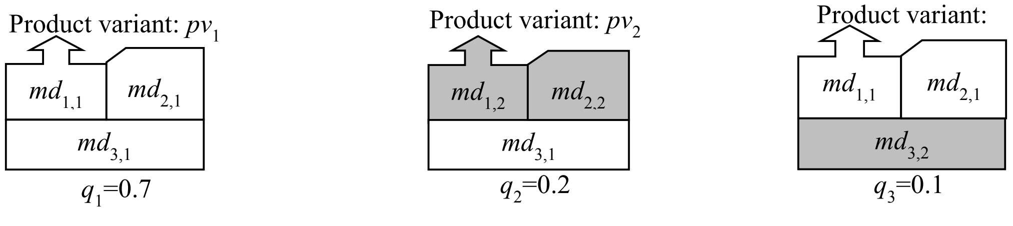


Figure 3-4 Mix ratio of product variants

Figure 3-5 shows a sample structure of supply chain network with its relevant mix ratio information. It shows that md_1 and md_2 (products of two suppliers, denoted as sp_1 and sp_2) are first assembled by a sub-assembler, and then the final assembler handles this sub-assembly and md_3 (product of sp_3) to finish the process. To record the mix ratios along the supply chain network, let $q_{i,v}$ be the mix ratio of the ith node of the supply chain network for producing the vth outputs. For instance, in Figure 3-5, the 5th node of the supply chain network is the final assembler, and its outputs correspond to the product variants in Figure 3-4 (i.e., $q_1 = q_{5,1}$, $q_2 = q_{5,2}$ and $q_3 = q_{5,3}$).

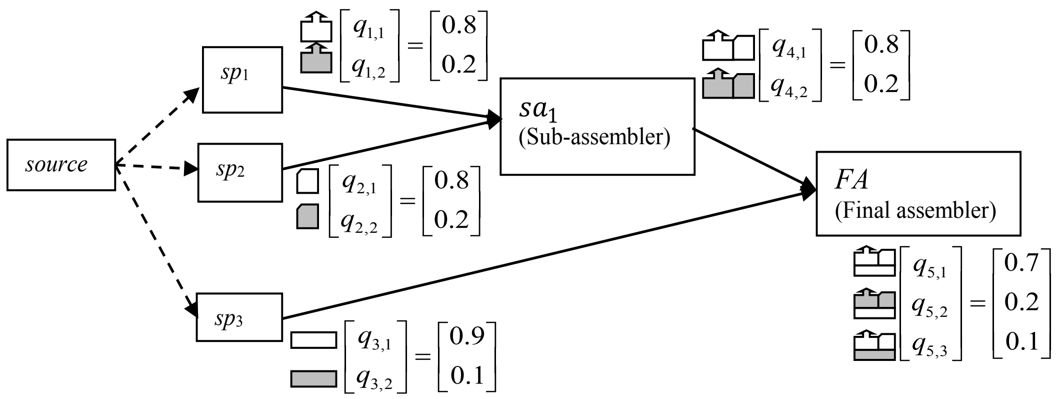

source
sp_1
sp_2
sp_3
sa_1
(Sub-assembler)
FA
(Final assembler)
$\begin{bmatrix} q_{1,1} \\ q_{1,2} \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$
$\begin{bmatrix} q_{2,1} \\ q_{2,2} \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$
$\begin{bmatrix} q_{3,1} \\ q_{3,2} \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$
$\begin{bmatrix} q_{4,1} \\ q_{4,2} \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$
$\begin{bmatrix} q_{5,1} \\ q_{5,2} \\ q_{5,3} \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$

Figure 3-5 A sample structure of supply chain network

As mentioned before, when the mix ratios of the product variants (output of final-assembler) are known, the mix ratios pertaining to other nodes in the supply chain network can be determined. Here, the basic problem is how to structure the supply chain network in view of forming the sub-assemblies. For example, by checking Figure 3-4, $md_{1,1}$ and $md_{2,1}$ are required by pv_1 and pv_3. Also, as pv_1 has a high demand, it should be reasonable to group $\{md_{1,1}, md_{2,1}\}$ as a sub-assembly in order to have a cost-effective production due to economy of scale.

Figure 3-5 shows the result of this arrangement that two sub-assemblies are produced and used in pv_1 and pv_3. Also in Figure 3-5, $q_{4,1} = 0.8$ corresponds to the sub-assembly $\{md_{1,1}, md_{2,1}\}$ for pv_1 and pv_3, and $q_{4,2} = 0.2$ corresponds to the sub-assembly $\{md_{1,2}, md_{2,2}\}$ for pv_2. Alternatively, if md_2 and md_3 are assembled first (instead of md_1 and md_2), three different sub-assemblies will be obtained: $\{md_{2,1}, md_{3,1}\}$ for pv_1, $\{md_{2,2}, md_{3,1}\}$ for pv_2 and $\{md_{2,1}, md_{3,2}\}$ for pv_3. At this point, the research inquiry is how to incorporate this analysis for structuring a supply chain network systematically.

3.3.2. Complexity model of a supply chain structure

To evaluate the goodness of a structure of supply chain network, the paper adopts the complexity measure developed by Wang et al. (2010). This complexity measure is based on the concept of information entropy. Let *CP* be the complexity measure, and its formulation is given as follows:

$$CP = log_2 e - \frac{1}{e}\sum_i e_i \sum_v q_{i,v}\, log_2\, q_{i,v} \tag{1}$$

Where e_i is the number of input edges of the ith supplier, and e is the total number of edges calculated by $e = \sum e_i$. Notably, the complexity measure is influenced by the structural configuration (i.e., e and e_i) and the mix ratios of suppliers (i.e., $q_{i,v}$). Lower complexity value means better structure of supply chain network. In brief, more edges and uniform mix ratio can lead to higher complexity. It means, when the firm offers more variants of the product and also more suppliers (sub-assembler) are engaged in the supply chain network, the level of complexity will be increased. Wang et al. (2009) investigated the relationship between complexity and cost of supply chain network. Their results reveal that complexity and cost following the same trend and agree with each other when comparing different structures of supply chain network with the same level of product variety.

In addition, less complex structure corresponds to the postponement strategy as the differentiating modules are not joined in the network until the final stage. In other words, if the differentiating module is grouped with other modules at the early stages of the network, we need to carry such complexity along the network which will lead to more complex structure. Hence, by postponing the assembly of the differentiation module to later stages of the network, we can avoid extra complexity of the supply chain network.

3.4. Methodology

3.4.1. Concept of coupling in supply chain network

The foundation of the proposed methodology is based on hierarchical clustering. Hierarchical clustering is a branch of cluster analysis that is considered as a systematic approach for forming groups or clusters (Everitt et al., 2011). Hierarchical clustering has been used as a solution approach for addressing product variety issues in manufacturing. Cell formation problem is one of the well-known applications of hierarchical clustering which has demonstrated in (McAuley, 1972). Relevant reviews can be found in the work of a Yin and Yasuda (2006). Also, Hölttä-Otto et al. (2008) have compared the commonality of modules based on their functional inputs / outputs and used the Euclidean distance and dendrogram for clustering the modules. Along the cell formation and hierarchical clustering, Navaei and ElMaraghy (2014) developed the commonality analysis by considering the operation requirement and machine options for grouping the products. Also, Kashkoush and ElMaraghy (2014) applied hierarchical clustering to develop components incidence matrix for forming product families with consideration of assembly sequence and demands.

Hierarchical clustering can be described in two major steps. The first step is to evaluate the similarity between any two objects. Since the term "similarity" may imply the consideration of common traits between two objects, this paper uses the term "coupling" for a more general concept. That is, coupling is referred to the measure of appropriateness to put two objects in a same group. If it is appropriate to group two objects together, the corresponding coupling value is high. Notably, appropriateness implies that the coupling value is application-dependent. That is, we may consider that objects a and b should be grouped in one case but not be grouped in another case.

Thus, the application context plays an important role to provide guidance in the formulation of coupling values. The second step of hierarchical clustering is tree construction which is performed by grouping objects progressively based on their coupling values.

In the context of structuring the supply chain network, the “coupling” question is to examine why two modules should be grouped together. For example, in Figure 3-5, should we group md_1 and md_2? If yes (or no), what is the reason? As any intermediate node (sub-assembler) in supply chain network will incur more edges (i.e. higher complexity), it is important for an intermediate sub-assembler to yield a “low-variety” mix ratio to counter the additional edges. It means, if two modules can be grouped to yield a “low-variety” mix ratio, the coupling value between them will be high.

Specifically, two modules are said coupled if their options are commonly selected by some product variant(s). In this reasoning, if two specific module options are often selected at the same time in many product variants, they should be grouped to form a sub-assembly that can be produced at high volume. Accordingly, the basic concept of the Jaccard coefficient (Everitt et al., 2011) is employed to evaluate the coupling values between modules options. Based on this concept, the detailed steps for calculating coupling values and constructing the supply chain network are developed in the next sub-section.

3.4.2. Methodical procedure

Step 1: Construct the product variety form

This step is to collect the product variety information and fill the product variety form. This form consists of two major parts. The first part is the binary product variety matrix that captures the relations between module options (e.g., $md_{1,1}$) and product variants (e.g., pv_1). Let $br_{i,j}$ be the binary matrix entry that $br_{i,j} = 1$ if the ith module option is selected in the jth product variant and $br_{i,j} = 0$ if ith module option is not selected in the jth product variant.

The second part of the product variety form contains the mix ratio information of module options and product variants that are placed along the corresponding rows and columns of the matrix. Figure 6 shows the product variety form where the gray area represents the product variety information according to the example in Figure 3-3. For normalization, the mix ratio of module options are divided by the number of modules and the normalized value of the ith module option is denoted as nq_i which is shows in the last column of Figure 3-6.

	pv_1 (mix ratio)	pv_2 (mix ratio)	pv_3 (mix ratio)	Mix ratio of module options	Normalized mix ratio
$md_{1,1}$	$br_{11} = 1$	$br_{12} = 0$	$br_{13} = 1$	$q_{1,1} = 0.80$	$nq_1 = 0.27$
$md_{1,2}$	$br_{21} = 0$	$br_{22} = 1$	$br_{23} = 0$	$q_{1,2} = 0.20$	$nq_2 = 0.07$
$md_{2,1}$	$br_{31} = 1$	$br_{32} = 0$	$br_{33} = 1$	$q_{2,1} = 0.80$	$nq_3 = 0.27$
$md_{2,2}$	$br_{41} = 0$	$br_{42} = 1$	$br_{43} = 0$	$q_{2,2} = 0.20$	$nq_4 = 0.07$
$md_{3,1}$	$br_{51} = 1$	$br_{52} = 1$	$br_{53} = 0$	$q_{3,1} = 0.90$	$nq_5 = 0.30$
$md_{3,2}$	$br_{61} = 0$	$br_{62} = 0$	$br_{63} = 1$	$q_{3,2} = 0.10$	$nq_6 = 0.03$
	$q_{5,1} = 0.70$	$q_{5,2} = 0.20$	$q_{5,3} = 0.10$		

Figure 3-6 Product variety form

Step 2: Determine the coupling values between module options

Based on the binary product variety matrix in product verity form, the basic concepts of the Jaccard coefficient is applied to compute the coupling values. Let $c_{i,j}$ be the coupling value between ith and jth module options and its formulation is given below:

$$c_{i,j} = \frac{\sum_{k=1}^{n} \min(br_{i,k}, br_{j,k})}{\sum_{k=1}^{n} \max(br_{i,k}, br_{j,k})} \tag{2}$$

Notably n is the total number of product variants. In this formulation, the min operator counts the number of product variants that use both ith and jth module options at the same time. The higher value for the min operation implies more product variants using these two specific module options, thus higher coupling values. Alternatively, the max operator in the denominator counts the total number of product variants that involve the ith and jth module options. This max operation is mainly used for normalization so that the coupling values are always kept between zero (no similarity) and one (highest similarity). Also, min and max operations can be considered as an alternative way to count 1-1 and 1-0 matches in the Jaccard coefficient. Figure 3-7 shows the square coupling matrix that captures the coupling values between module options of the example.

	$md_{1,1}$	$md_{1,2}$	$md_{2,1}$	$md_{2,2}$	$md_{3,1}$	$md_{3,2}$
$md_{1,1}$		0	1	0	0.33	0.50
$md_{1,2}$	0		0	1	0.50	0
$md_{2,1}$	1	0		0	0.33	0.50
$md_{2,2}$	0	1	0		0.50	0
$md_{3,1}$	0.33	0.50	0.33	0.50		0
$md_{3,2}$	0.50	0	0.50	0	0	

Figure 3-7 Coupling matrix of module options

Step 3: Adjust the coupling values based on mix ratios

In this step the coupling values are adjusted according to the mix ratio of two module options. Basically, higher mix ratio indicates higher demand for a specific module option (product variants). Hence, if the mix ratio of two module options is high, the grouping of these module options can potentially reduce the complexity due to high demand for specific product variants.

Thus, the coupling values in the previous step are adjusted based on the information of mix ratios. Let $ac_{i,j}$ be the adjusted coupling value between ith and jth module options and the formulation is given below:

$$ac_{i,j} = c_{i,j}(nq_i + nq_j) \tag{3}$$

As a result, when demand for module option i and module option j is high, it increases the coupling value between them in order to encourage the grouping of these two module options in the structuring process. Figure 3-8 shows the adjusted coupling values.

	$md_{1,1}$	$md_{1,2}$	$md_{2,1}$	$md_{2,2}$	$md_{3,1}$	$md_{3,2}$
$md_{1,1}$		0	***0.54***	***0***	0.19	0.15
$md_{1,2}$	0		***0***	***0.14***	0.19	0
$md_{2,1}$	0.54	0		0	0.19	0.15
$md_{2,2}$	0	0.14	0		0.19	0
$md_{3,1}$	0.19	0.19	0.19	0.19		0
$md_{3,2}$	0.15	0	0.15	0	0	

Figure 3-8 Adjusted coupling matrix

Step 4: Determine the coupling values between modules

In the process of structuring the supply chain network, the grouping process is carried out over the modules rather than module options. Therefore, this step of the process is to determine the coupling values between two modules by considering all relevant variants of a module. For example, in Figure 3-8, by averaging four coupling values between md_1 and md_2 (**bolded** and *italicized*), we can find the coupling value between md_1 and md_2 that is reflected in Figure 3-9.

	md_1	md_2	md_3
md_1		***0.17***	0.13
md_2	0.17		0.13
md_3	0.13	0.13	

Figure 3-9 Coupling matrix between modules

Step 5: Construct the supply chain network

Based on the coupling matrix in Figure 3-9, the standard procedure of hierarchical clustering is applied to construct the tree. The resulting tree for the example is presented in Figure 3-10 that the top branch represents the position of final assembler. Then, the tree structure basically suggests how different modules should be grouped together leading to final assembler. Two different structures that are suggested by different cut lines are demonstrated in Figure 3-11.

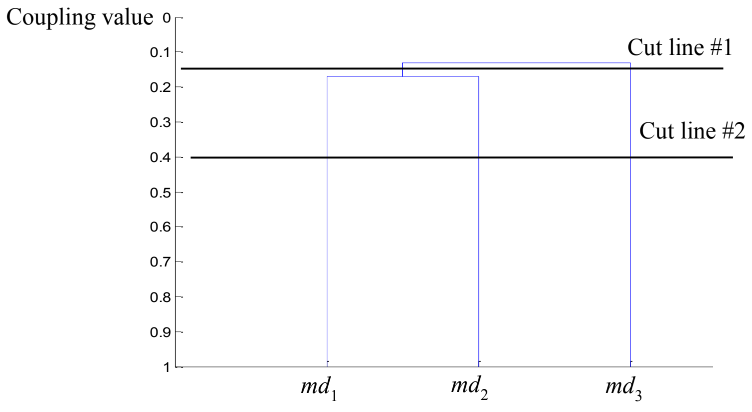


Figure 3-10 Tree-based analysis

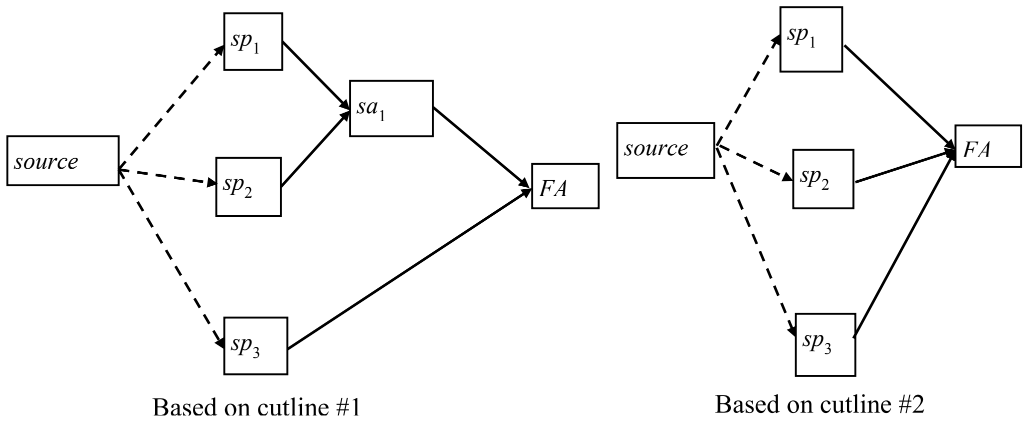


Figure 3-11 Two possible structures of supply chain network

There are two guidelines to construct supply chain network based on tree results. The first guideline is based on tree cutting. Consider two cut lines in Figure 3-10, if cut line #1 is applied, two branches are cut, indicating that final assembler receives two sub-assemblies, one from md_1 and md_2 and another from md_3. Likewise, if the cut line #2 is applied, three branches are cut and a non-modular structure of supply chain network is obtained. In this guideline, the position of the cut line reflects the threshold of the coupling values to define the grouping condition. In the example tree in Figure 3-10, cut line #1 imposes a lower threshold value and therefore final assembler expects to perform less number of assemblies (more sub-assemblers). Alternatively, cut line #2 indicates a higher threshold value leading to more assembly operation at the final assembler.

The second guideline is based on the joint point proximity which is concerned with whether a subset of modules should be grouped in one tier or separated in two tiers. For example, in sample example, as the joint point of md_1 and md_2 are relatively close to joint point of ($md_1 md_2$) and md_3, these modules can form a one tier network structure due to proximate join points and also the low coupling value threshold.

However, these two guidelines are intended to provide some basic information for using tree-based analysis to construct the supply chain network. Automated solution processes are postponed to future works since they need to be customized according to contextual details in the analysis. From the algorithmic view, the key benefit of using tree-based analysis is to narrow down the possible solutions in the grouping process. Even for the small sample example, as the tree-based analysis suggests the grouping of md_1 and md_2, other possible groupings (e.g., grouping md_1 and md_3) are eliminated from further considerations.

3.5. Numerical Examples

To examine the applicability and effectiveness of the proposed method, five numerical examples (cases) are set. Case #1 and Case #2 are demonstrative cases that are purposely designed to test the capability of the proposed method to detect the structure of supply chain network with lower complexity value in the extreme cases. Case #3 and Case #4 are designed to show the capability of the method to find the supply chain network with lower complexity value. Case #5 is intended to demonstrate the advantages of the proposed method in terms of supply chain practices like postponement.

3.5.1. Discussion of Case #1 and Case #2

The product variety forms of Case #1 and Case #2 as the input of the methodical procedure are provided in Figure 3-12. Case #1 is a printer that has four modules namely print cartridge, paper tray, printer case and power adaptor. Suppose this product is offered in different geographical locations. Considering the case of different electrical power requirements for the product in different regions, the power adaptor is offered in two options (e.g., 110 VAC and 220 VAC). Case #2 is a product composed of three modules, and each module has two variants leading to eight different product variants with the same mix ratio.

Figure 3-13 shows the resulting tree and the supply chain network for sample printer in Case #1. As it can be seen, the power adaptor (md_4) is not joined with other modules until the end of the process. That is, the process of assembling power adaptor is postponed until determining the geographical location of the order. This postponement strategy enables inventory cost reduction as there is no need to keep stock of two kinds of printers.

		pv_1	pv_2	
Print cartridge	$md_{1,1}$	1	1	1
Paper tray	$md_{2,1}$	1	1	1
Printer case	$md_{3,1}$	1	1	1
Power adaptor 110 VAC	$md_{4,1}$	1	0	0.5
Power adaptor 220 VAC	$md_{4,2}$	0	1	0.5
		0.5	0.5	

Case #1

	pv_1	pv_2	pv_3	pv_4	pv_5	pv_6	pv_7	pv_8	
$md_{1,1}$	1	1	1	1	0	0	0	0	0.5
$md_{1,2}$	0	0	0	0	1	1	1	1	0.5
$md_{2,1}$	1	1	0	0	1	1	0	0	0.5
$md_{2,2}$	0	0	1	1	0	0	1	1	0.5
$md_{3,1}$	1	0	1	0	1	0	1	0	0.5
$md_{3,2}$	0	1	0	1	0	1	0	1	0.5
	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	

Case #2

Figure 3-12 Product variety forms of Case #1 and Case #2

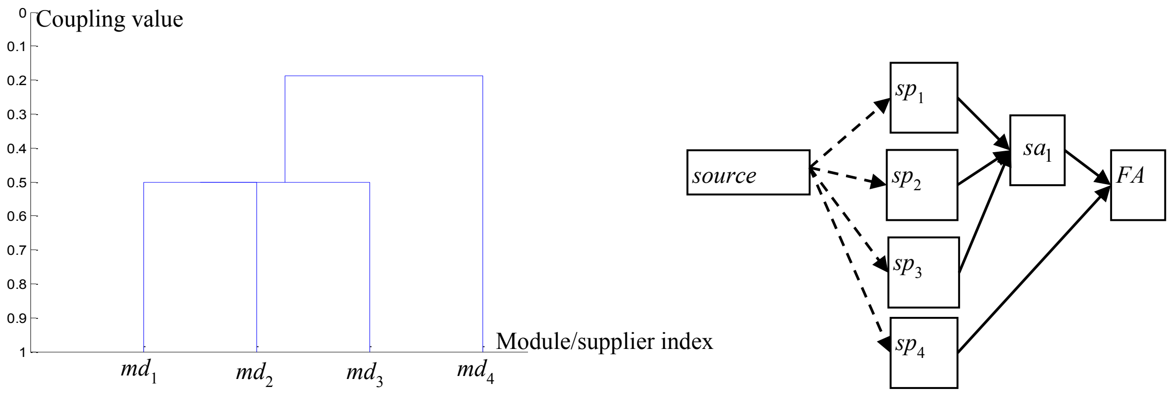


Figure 3-13 Tree result and supply chain network of printer example (Case #1)

Figure 3-14 shows the tree result and the supply chain network for Case #2. As it can be seen in this case, there is no differentiating module to postpone and all the modules and product variants have same mix ratio. That is, there is no module or product variant that has privilege over others. Therefore, non-modular structure of supply chain network is obtained in this case that final assembler is responsible for assembly of all modules.

To represent the supply chain network compactly, the "bracket" representation from Wang et al. (2010) is adapted. For example, the supply chain networks for Case #1 and Case #2 are represented as $((sp_1, sp_2, sp_3)(sp_4))$ and (sp_1, sp_2, sp_3) respectively, that the brackets indicate the grouping order of modules.

For comparison, the complexity values of modular and non-modular networks are determined for Case #1 and Case #2, and the values are recorded in Table 3-1. Also, the complexity values based on the proposed method are highlighted in grey color. As seen, the proposed method can suggest the networks of lower complexity values. Yet, whether a manager should employ a modular or non-modular structure of supply chain network is a situational issue. The proposed method in this paper provides one tool for the relevant analysis.

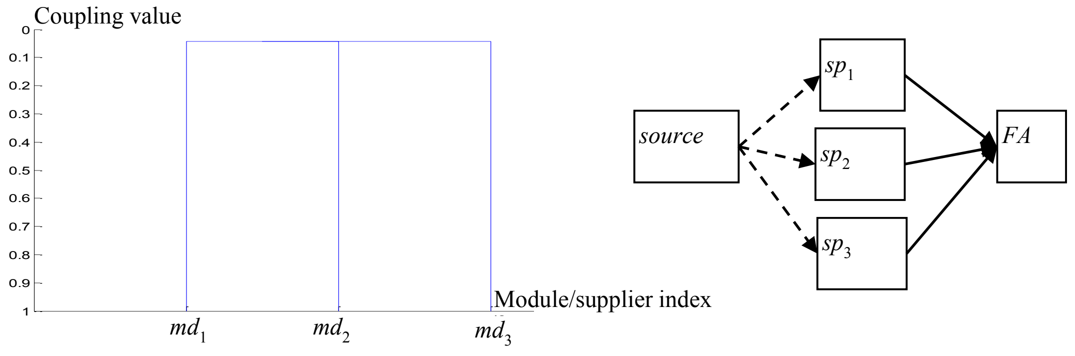


Figure 3-14 Tree result and supply chain network of case #2

Table 3-1 complexity measure for Case #1 and Case #2

	Supply Chain Network		**Complexity**
Case #1	$((sp_1, sp_2, sp_3)(sp_4))$	Modular	3.503
	(sp_1, sp_2, sp_3, sp_4)	Non-modular	3.625
Case #2	$((sp_1, sp_2)\ (sp_3))$	Modular	4.664
	(sp_1, sp_2, sp_3)	Non-modular	4.585

3.5.2. Discussion of Case #3 and Case #4

Case #3 and Case #4 are based on a product that has four modules leading to five product variants in total. For investigation purpose, there are no "dominating" module options that are frequently used by product variants in Case #3. Alternatively, $md_{1,1}$ and $md_{2,1}$ are purposely set to be used in most of product variants in Case #4. Figure 3-15 shows the product variety forms for Cases #3 and Case #4.

	pv_1	pv_2	pv_3	pv_4	pv_5	
$md_{1,1}$	1	1	0	0	0	0.27
$md_{1,2}$	0	0	1	0	0	0.18
$md_{1,3}$	0	0	0	1	1	0.55
$md_{2,1}$	1	0	0	1	1	0.66
$md_{2,2}$	0	1	1	0	0	0.34
$md_{3,1}$	0	0	0	1	0	0.31
$md_{3,2}$	0	1	1	0	0	0.34
$md_{3,3}$	1	0	0	0	1	0.35
$md_{4,1}$	0	1	1	1	0	0.65
$md_{4,2}$	0	0	0	0	1	0.24
$md_{4,3}$	1	0	0	0	0	0.11
	0.11	0.16	0.18	0.31	0.24	

Case #3

	pv_1	pv_2	pv_3	pv_4	pv_5	
$md_{1,1}$	1	1	1	1	1	1
$md_{2,1}$	1	1	1	1	0	0.89
$md_{2,2}$	0	0	0	0	1	0.11
$md_{3,1}$	1	1	1	0	1	0.69
$md_{3,2}$	0	0	0	1	0	0.31
$md_{4,1}$	1	0	0	1	1	0.66
$md_{4,2}$	0	1	0	0	0	0.16
$md_{4,3}$	0	0	1	0	0	0.18
	0.24	0.16	0.18	0.31	0.11	

Case #4

Figure 3-15 Product variety forms of Case #3 and Case #4

Figure 3-16 shows the tree and supply chain network for Case #3. The tree structure basically suggests how different modules should be grouped together leading to final assembler. Consider that the cut line is set at 0.1. This line basically cuts all four branches, reflecting that final assembler handles all the assemblies without having sub-assemblers.

Figure 3-17 shows the tree and the supply chain network for Case #4. In this case, if the cut line is set at 0.1, we will obtain a modular structure of supply chain network that groups md_1 and md_2 and then with md_3, reflecting that final assembler handles two sub-assemblies. Briefly, this supply chain network seems reasonable because $md_{1,1}$ and $md_{2,1}$ are used by most of product variants with high mix ratio of 0.89.

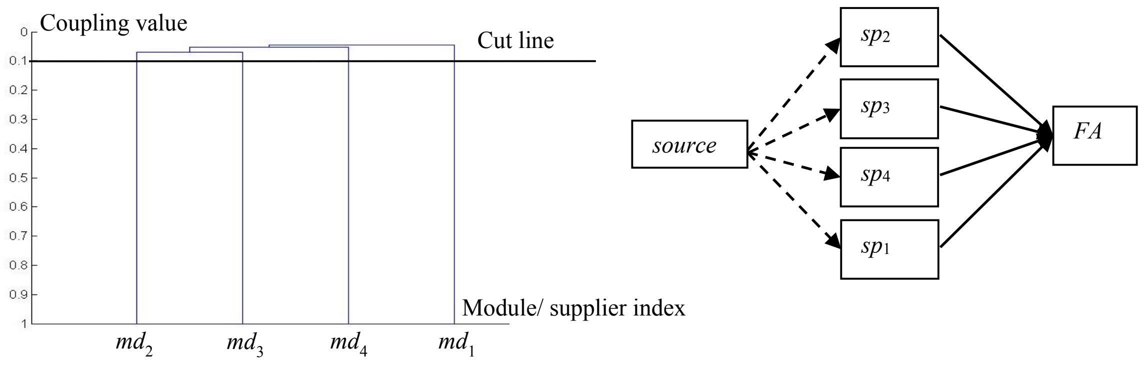


Figure 3-16 Tree construction and supply chain network of Case #3

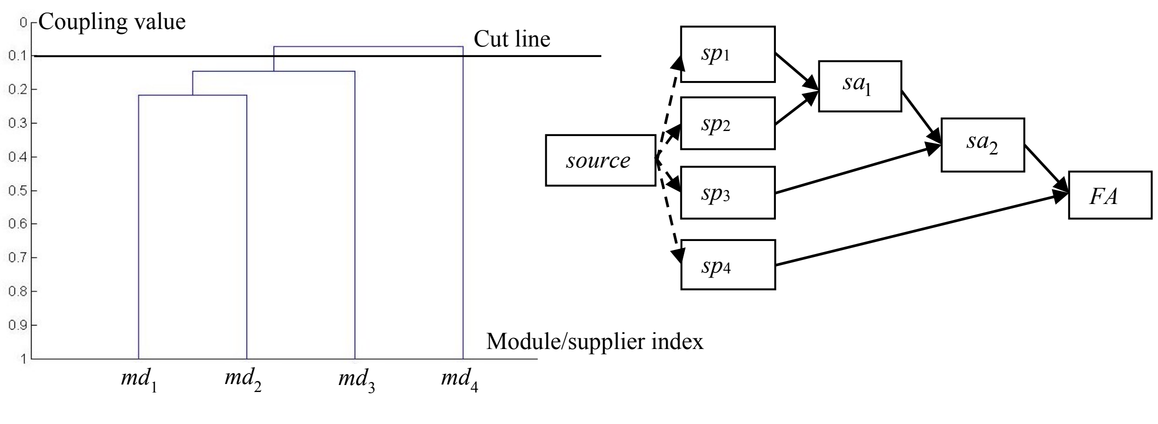


Figure 3-17 Tree construction and supply chain network of Case #4

To further verify the quality of the supply chain networks obtained by the proposed method, all possible supply chain networks are exhaustively determined along with their complexity values in Table 3-2. The complexity values of the supply chain networks suggested for Case #3 and Case #4 are highlighted by grey color. As seen in Table 3-2, the proposed method can suggest the lowest complexity values. These results support the utility of the proposed hierarchical clustering approach towards the complexity reduction of supply chain network.

Table 3-2 Exhaustive comparison of complexity measures in Case #3 and Case #4

Supply Chain Network	**Complexity**		**Supply Chain Network**	**Complexity**	
	Case #3	**Case #4**		**Case #3**	**Case #4**
(sp_1, sp_2, sp_3, sp_4)	4.767	4.450	$((sp_3, sp_4)\ (sp_1)\ (sp_2))$	4.913	4.638
$((sp_1, sp_2, sp_3)\ (sp_4))$	4.989	4.405	$(((sp_1, sp_2)\ (sp_3))\ (sp_4))$	5.074	4.401
$((sp_2, sp_3, sp_4)\ (sp_1))$	4.876	4.708	$(((sp_1, sp_2)\ (sp_4))\ (sp_3))$	5.074	4.474
$((sp_1, sp_3, sp_4)\ (sp_2))$	4.989	4.603	$(((sp_1, sp_3)\ (sp_2))\ (sp_4))$	5.182	4.480
$((sp_1, sp_2, sp_4)\ (sp_3))$	4.989	4.526	$(((sp_1, sp_3)\ (sp_4))\ (sp_2))$	5.182	4.598
$((sp_1, sp_2)\ (sp_3, sp_4))$	5.006	4.519	$(((sp_1, sp_4)\ (sp_2))\ (sp_3))$	5.182	4.626
$((sp_1, sp_3)\ (sp_2, sp_4))$	5.115	4.552	$(((sp_1, sp_4)\ (sp_3))\ (sp_2))$	5.182	4.672
$((sp_1, sp_4)\ (sp_2, sp_3))$	5.052	4.554	$(((sp_2, sp_3)\ (sp_1))\ (sp_4))$	5.052	4.567
$((sp_1, sp_2)\ (sp_3)\ (sp_4))$	4.868	4.322	$(((sp_2, sp_3)\ (sp_4))\ (sp_1))$	4.984	4.748
$((sp_1, sp_3)\ (sp_2)\ (sp_4))$	4.989	4.409	$(((sp_2, sp_4)\ (sp_1))\ (sp_3))$	5.115	4.712
$((sp_1, sp_4)\ (sp_2)\ (sp_3))$	4.989	4.492	$(((sp_2, sp_4)\ (sp_3))\ (sp_1))$	5.047	4.821
$((sp_2, sp_3)\ (sp_1)\ (sp_4))$	4.843	4.506	$(((sp_3, sp_4)\ (sp_1))\ (sp_2))$	5.115	4.804
$((sp_2, sp_4)\ (sp_1)\ (sp_3))$	4.913	4.587	$(((sp_3, sp_4)\ (sp_2))\ (sp_1))$	5.047	4.867

3.5.3. Discussion of Case #5

In order to demonstrate the applicability of the method to support postponement, the product variants based on the information from a car manufacturing company are considered. Particularly, nine modules are identified to offer five different product variants. The product variety form is provided Figure 3-18.

			pv_1	pv_2	pv_3	pv_4	pv_5	
Engine	2.5 L	$md_{1,1}$	1	1	1	1	1	1
Drive	FWD	$md_{2,1}$	1	0	1	0	0	0.4
	AWD	$md_{2,2}$	0	1	0	1	1	0.6
Trans	Manual	$md_{3,1}$	1	0	0	0	0	0.2
	5-Speed	$md_{3,2}$	0	1	1	1	1	0.8
Wheel	16-inch	$md_{4,1}$	1	1	0	0	0	0.4
	17-inch	$md_{4,2}$	0	0	1	1	1	0.6
	18-inch	$md_{4,3}$	0	0	0	0	1	0.2
Cargo	Manual	$md_{5,1}$	1	1	1	1	1	1
Door	Key	$md_{6,1}$	1	1	1	1	0	0.8
	Keyless	$md_{6,2}$	0	0	0	0	1	0.2
Seat	Cloth	$md_{7,1}$	1	1	1	1	0	0.8
	Leather	$md_{7,2}$	0	0	0	0	1	0.2
Climate	Single	$md_{8,1}$	1	1	0	0	0	0.4
	Dual	$md_{8,2}$	0	0	1	1	1	0.6
Audio	4-Speakers	$md_{9,1}$	1	1	0	0	0	0.4
	6-Speakers	$md_{9,2}$	0	0	1	1	0	0.4
	8-Speakers	$md_{9,3}$	0	0	0	0	1	0.2
			0.2	0.2	0.2	0.2	0.2	

Figure 3-18 Product variety form of Case #5

Given the product variety form, the supply chain structuring problem is how to group the modules and assign them to suppliers. Although it is not a very large case, it is still difficult to manage the relationship among different module options in order to bundle them in groups. In this way, the proposed method can help supply chain managers to determine the structure of supply chain network and manage product variety issue in terms of supporting the postponement strategy.

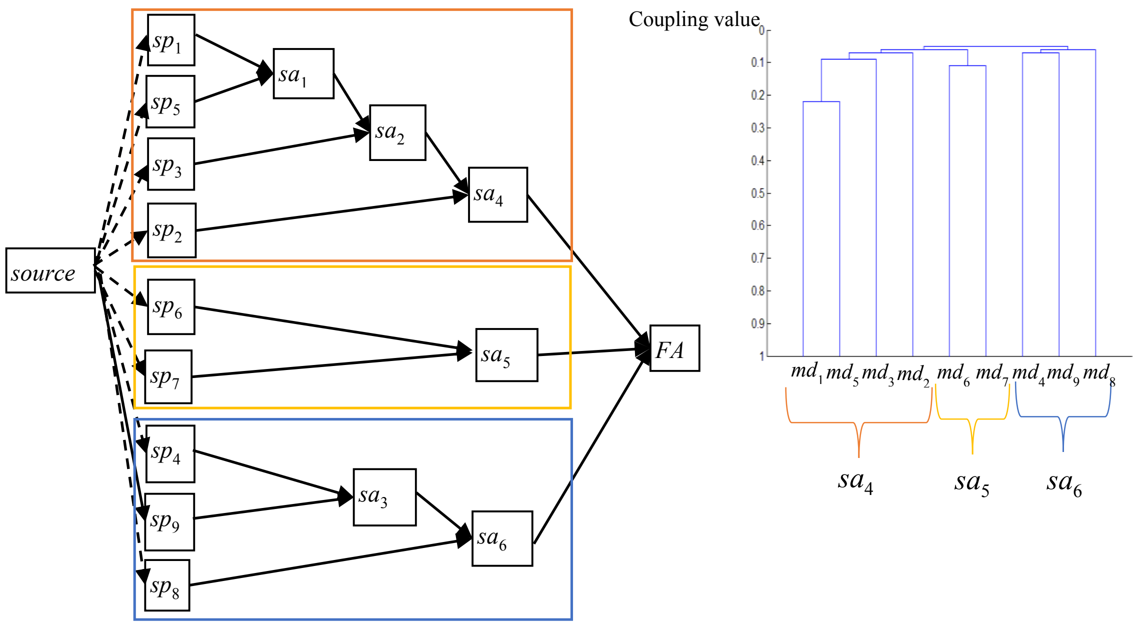


Figure 3-19 Supply chain network of Case #5

Figure 3-19 shows the supply chain network of Case #5. As seen, a modular supply chain network is suggested that final assembler is responsible for assembly of three major sub-assemblies that are produced by three major sub-assemblers (i.e., sa_4,sa_5,sa_6). Multiple tiers are also found in two sub-assemblers. In sa_4, the transmission (md_3) is the differentiating module that will be delayed till the end of the assembly process, and in sa_6, the audio system (md_9) is the differentiating module. In contrast, there is no differentiating module in sa_5. Notably, this kind of supply chain network needs close relationships between sub-assemblers and final assembler in terms of transmitting the demand information to the sub-assemblers so that the right choice of the differentiating module can be made.

The supply chain network in Figure 3-19 can help supply chain managers to manage product variety in two ways. Firstly, the modular supply chain network shows the number of suppliers/sub-assemblers that are required in the process and how the product should be built upon reaching final assembler and delivering to customers. Secondly, they can identify which modules are “similar” for supporting managerial decisions. For example, the module options of md_6 and md_7 are correlated that $md_{6,1}$ and $md_{7,1}$ are always selected in the same product variants, so do $md_{6,2}$ and $md_{7,2}$. Therefore, by grouping these two modules in the supply chain network, further decisions (i.e. production, procurement and inventory) regarding these two modules can be aggregated in order to support the product variety management.

3.6. Closing Remarks

In support of product variety, this paper proposed a methodical procedure for constructing the supply chain network. The proposed method is based on hierarchical clustering, and the core technique is to analyze the coupling values between the product’s modules according to the product variety information such as product variants and mix ratios. Five cases have been developed, and the complexity values of various structures of supply chain network have been calculated to demonstrate that the proposed method can help to yield supply chain networks with low complexity values. Moreover, the application of the proposed method in terms of managing the postponement strategy has been discussed. It is demonstrated that the proposed method is capable to identify and postpone differentiating modules to downstream processes in supply chain.

In general, hierarchical clustering is a reliable approach for solving problems with grouping decisions. The coupling analysis quantifies the appropriateness of grouping two objects together. Since hierarchical clustering is not an exhaustive search algorithm, the optimal solution cannot be guaranteed. Yet, the method can effectively yield reasonable solutions in a limited time.

In future work, it is planned to broaden the application of hierarchical clustering for addressing product variety problems. Specifically, hierarchical clustering can be applied for solving some structural or network problems like vanilla box configuration or effects of commonality planning on procurement policy of supply chain.

3.7. Author's Notes and Significance of Paper to Thesis

In this chapter, one crucial input information for structuring supply chain network is the modular structure of the product that has been identified in the previous chapter. It shows the way that defining modular structure of the product affects its supply chain network. That is, defining more solid and independent modules during product architecture design will help to have less complex supply chain network. In the reasoning, postponement strategy plays a key role and the fact that more independent modules facilitate postponement strategy.

Beyond structuring supply chain network, the proposed method can be applied to similar problems like assembly supply chain problem in order to minimize the complexity of the structure. The paper concerning the application of the proposed method in assembly supply chain can be found in Appendix A.

At the end, while not criticizing optimization approaches, the efforts in this paper and also assembly supply chain paper (in Appendix A) can prove that the proposed method based on hierarchical clustering can be an alternative solution approach for solving such grouping problems when the optimality is not the main concern. As the grouping problems are often NP-complete, globally optimal solutions are not easy to obtain anyway. Thus, hierarchical clustering is not an unreasonable choice for problem solving.

4. Managing Product Variety through Configuration of Pre-assembled Vanilla Boxes Using Hierarchical Clustering

ABSTRACT

Postponement strategy and platform-based production are common practices of mass customization to address supply chain challenges due to the requirement of product variety. This paper focuses on implementing mass customization through development of semi-finished products (vanilla boxes) to reduce supply chain cost and facilitate the production process. The challenge is that the possible number of vanilla box configurations grows dramatically with the increase in number of product variants. In the solution approach, the basic information of product variety is captured in a matrix format, specifying the component requirements for each product variant. Then, hierarchical clustering is applied over the components with the considerations of demands. The clustering method consists of three major stages: similarity analysis, tree construction and tree-based analysis. The key stage is similarity analysis, in which problem-specific information can be incorporated in the clustering process. Two numerical examples from the literature are used to verify that the clustering approach can yield good quality solutions.

4.1. Introduction

To satisfy customer needs and survive in today's competitive market, manufacturers move towards mass customization to offer higher level of product variety and shorter lead time (Venkatesh and Swaminathan, 2004). Mass customization can be defined as producing products to satisfy individual customer's needs with near mass production efficiency (Parlaktrk and Swaminathan, 2010). In this context, the traditional make-to-stock policy may not be the best supply chain strategy since the uncertain demand of individual product variants can potentially lead to high inventory costs and stock-out losses. Therefore, make-to-order approaches are often considered for supply chain of customized product.

In order to support the make-to-order approach and tackle challenges due to high product variety, one well-known technique is postponement (ElMaraghy et al., 2013). The basic idea behind postponement is that the risk and uncertainty are tied with differentiating parts of the product. Therefore, to the extent that manufacturing and operation of these parts can be postponed, the risk, uncertainty and complexity of manufacturing can be reduced (Kim, 2014).

Along with postponement, platform-based product development has been well recognized as an effective means to control the production cost with increasing demands for product variety. Using platform-based manufacturing, every product variants can either be assembled directly from its components, or from a platform that has already designed and its components resemble those required by the product (Jiao et al., 2007). The basic idea behind platform-based manufacturing is to reduce assembly operations in make-to-order supply chain by using some semi-finished products. In this way, shorter lead times can be achieved by holding inventory of these semi-finished products. Swaminathan and Tayur (1998) denote these semi-finished products as vanilla boxes in the context of computer industry.

In this paper, we address the Vanilla Box Configuration (VBC) problem that is concerned with finding the groups of components (i.e. vanilla boxes) that are common to some product variants in order to reduce cost. The combinatorial nature of the problem makes the possible solution space grow dramatically with the number of components.

Due to the NP-complete nature of the VBC problem (Dawande et al., 2001), the effectiveness of the solution approach should be examined via the solution quality as well as the practicality of algorithms (e.g., whether it is computationally complex and takes long time to run). In this paper, hierarchical clustering is proposed as a practical tool for solving the VBC problem in the management of product variety. Specifically, the proposed approach has advantages in terms of its algorithmic simplicity and flexibility to incorporate new factors in forming of vanilla boxes. This will be demonstrated through two numerical examples in Section 4.5. While the first example (in Section 4.5.1) demonstrates the applicability of hierarchical clustering as a solution approach for solving VBC problems, the second example (in Section 4.5.2) demonstrates the flexibility of the proposed method in incorporating a new factor in the solution approach.

The rest of paper is organized as follows. In Section 4.2, we review the relevant literature. Section 4.3 defines the problem, and describes basic information of product variety. The basic procedure of hierarchical clustering used in this paper is provided in Section 4.4. Section 4.5 presents two numerical examples to show the application of hierarchical clustering for solving VBC problems. Section 4.6 provides conclusions.

4.2. Literature Review

We consider the work by Swaminathan and Tayur (1998) as a starting point for relevant decisions regarding VBC in our research problem. This problem has several similarities with product platform development problem (Krishnan and Ulrich, 2001, Simpson, 2004). Beyond the contextual details, these two concepts are quite similar in terms of forming component groups by considering design functions (i.e. ‘module” or “platform” concept) or supply chain operations (i.e. “vanilla box” concept). For example, the multi-platform configuration problem (Ben-Arieh et al., 2009), modular platform (Gershenson et al., 2004) and the modular design problem (Agard and Bassetto, 2013) actually investigates the formation of component groups with various considerations (production cost, quality). Therefore, like modular platform that can be used for different functions by adding, removing or substituting modules (Li et al., 2013); the same concept is applicable for vanilla boxes as they can be used for production of different products through the addition, substitution and/or exclusion of components.

Besides, when vanilla boxes are considered as semi-finished form of products, they become essential for determining the “point of differentiation” (Al-Salim and Choobineh, 2009, Wong et al., 2009, Lu et al., 2012). Therefore, the problem of VBC can be relevant to some issues of product variety such as product platform and product differentiation points, and its solution approach can be quite fundamental for the application of postponement.

Moreover, the VBC problem can be characterized as a maximum edge cardinality biclique problem which is NP-complete (Dawande et al., 2001, Peeters, 2003). Thus, various meta-heuristic approaches were implemented for solving similar problems including genetic algorithm (Ben-Arieh et al., 2009), simulated annealing (Agard and Bassetto, 2013) and tabu search (Al-Salim and Choobineh, 2009).

Compared with configuration efforts in VBC problems, the application of hierarchical clustering as the solution approach is relatively limited in the literature. This paper is intended to contribute in this research direction and propose a method based on hierarchical clustering.

4.3. Background

4.3.1. Vanilla box configuration problem

For managing product variety, one basic piece of information is a set of product variants. Let $PV = \{pv_1, pv_2, pv_3, \ldots, pv_m\}$ be the set of m product variants. For production of these product variants, a set of components is required. Let $C = \{c_1, c_2, c_3, \ldots, c_n\}$ be the set of n components that are required for producing all product variants. Several product variants can share one or more common components. To capture such information, the product variety matrix is applied and formulated as $M = [m_{i,j}]$. Particularly, M is a binary matrix that $m_{i,j} = 1$ if the ith product variant requires the jth component (else, $m_{i,j} = 0$).

For example, Figure 4-1 represents a product family with five different product variants (pv_1, pv_2, pv_3, pv_4 and pv_5) that has been designed according to the pilot study example from Swaminathan and Tayur (1998). Figure 4-2a shows the product variety matrix (i.e., *M*) for the sample product family. In this context, a vanilla box is a subset of components. For example, suppose that c_4 and c_5 form a vanilla box, which can be used to produce pv_1, pv_2, pv_3 and pv_5 by adding components (e.g., adding c_1 to this vanilla box gives pv_1). In this case, the vanilla box $\{c_4, c_5\}$ can be potentially prepared with high volume for less cost.

Let $V = \left\{v_1, v_2, \ldots, v_k, \ldots, v_p\right\}$ be the set of p vanilla boxes, where $v_k \subseteq C$. Then, the vanilla box configuration (VBC) problem is to determine V so that the product variants can be produced from these vanilla boxes with minimal addition and removal of components. In practice, as each vanilla box incurs a setup cost, it may not be economical to have many small vanilla boxes. Alternately, a large vanilla box can make it less useful for most product variants. It turns out the "balanced" configuration of vanilla boxes is not a trivial problem. The basic inquiry is to group components that are mainly shared by some product variants in a vanilla box. In this way, each vanilla box can be used to make product variants with minimum changes (i.e., adding or removing components) in order to justify the setup cost.

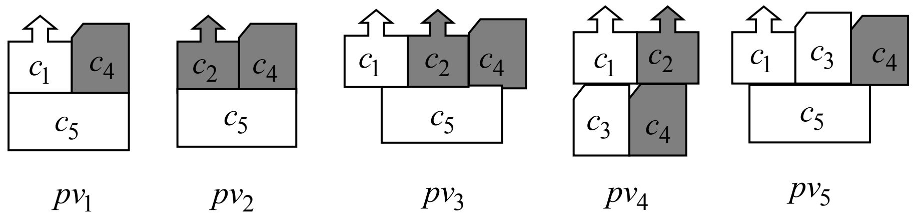


Figure 4-1 Example of product variants

	c_1	c_2	c_3	c_4	c_5
pv_1	1	0	0	1	1
pv_2	0	1	0	1	1
pv_3	1	1	0	1	1
pv_4	1	1	1	1	0
pv_5	1	0	1	1	1

4-2a. *sample of product variety matrix*

	c_1	c_2	c_3	c_4	c_5
c_1	-	0.40	0.50	0.80	0.60
c_2	0.40	-	0.25	0.60	0.40
c_3	0.50	0.25	-	0.40	0.20
c_4	0.80	0.60	0.40	-	0.80
c_5	0.60	0.40	0.20	0.80	-

4-2b. *Similarity matrix of components (SC)*

Figure 4-2 sample product variety matrix and its relevant similarity matrix

4.3.2. Hierarchical clustering

Hierarchical clustering is a branch of cluster analysis that is considered as a systematic approach for forming groups or clusters (Everitt et al., 2011). Hierarchical clustering has been used as a solution approach for addressing product variety issues in manufacturing. Cell formation problem is one of the well-known application of hierarchical clustering (Yin and Yasuda, 2006).

In hierarchical clustering, one crucial step in the clustering process is to evaluate the similarity between any two objects. The clustering process proceeds by progressively grouping the objects that share high similarity values with each other. In this paper, similarity is interpreted as the appropriateness to group two objects in one application. Thus, the similarity value is application-dependent. That is, we may consider that objects a and b should be grouped together in one case but not grouped in another case. Then the application context becomes the important information to guide the formulation of similarity values.

4.3.3. Remarks

Computationally, the difficulty of VBC problem stems from the large number of possible vanilla boxes as result of grouping components. For example, the number of vanilla boxes with five components and four end products can be as large as 22 without considering vanilla boxes with zero components, one component and vanilla boxes with all components (Kuthambalayan et al., 2014). To tackle this difficulty, one intuitive approach is to group highly linked components and identify the structure of the groups. This idea motivates the use of hierarchical cluttering as the solution approach.

While the clustering process of this paper stays with the traditional average linkage method (Everitt et al., 2011), the new element is to develop the similarity analysis for VBC problem. Particularly, this paper demonstrated how the similarity values can be adjusted according to the specific information of the problem context (i.e. demand information). To verity the quality of solutions from hierarchical clustering, two examples from the literature (in Section 4.5) are purposely used for comparison study.

4.4. Hierarchical Clustering Methodology

In this paper, the basic computational problem is to define the groups of components using the information from the product variety matrix. The foundation of the methodical approach is based on hierarchical cluster analysis (Everitt et al., 2011) that is described in two steps in this section. The first step is similarity analysis, which evaluates appropriateness of grouping two objects together. The second step is tree construction, which is performed by grouping the objects progressively based on their similarity values. Afterwards, the tree-based analysis is discussed to show how to use the tree information for addressing relevant product variety problems.

4.4.1. Similarity analysis

In the context of VBC, the key question of similarity analysis is whether two components should be grouped together and to explain why. In this work, two components are considered similar if they are selected in the same product variant(s). In the reasoning, if two components are often used in many product variants, they should be grouped to form a vanilla box that can be produced in high volumes. Based on the product variety matrix (M), the concept of Jaccard coefficient is employed to evaluate the similarity values. Let sc_{ij} be the similarity value between the ith and jth components and its formulation is given below:

$$sc_{ij} = \frac{\sum_{k=1}^{m} \min(m_{ki}, m_{kj})}{\sum_{k=1}^{m} \max(m_{ki}, m_{kj})} \tag{1}$$

In this formulation, the *min* operation in the numerator counts the number of product variants that use both ith and jth components at the same time. The higher values for the *min* operation imply more product variants using these two specific components, thus higher similarity values. Alternatively, the *max* operation in the denominator counts the total number of product variants that involve the ith or jth components. This *max* operation is mainly used for normalization so that the similarity values are always kept between zero (no similarity) and one (highest similarity). Also, *min* and *max* operations can be considered as an alternative way to count 1-1 and 1-0 matches in the Jaccard coefficient.

To illustrate, consider product variants in Figure 4-1 and its relevant product variety matrix in Figure 4-2a. The similarity value between c_1 and c_2 is equal to 0.4 as they are commonly found in two product variants (i.e. pv_3 and pv_4) and all five product variants have at least one of them. Figure 4-2b shows the resulting similarity matrix, denoted as $SC = [sc_{ij}]$. The similarity matrix SC records the similarity values between any two components, and it is a symmetric matrix.

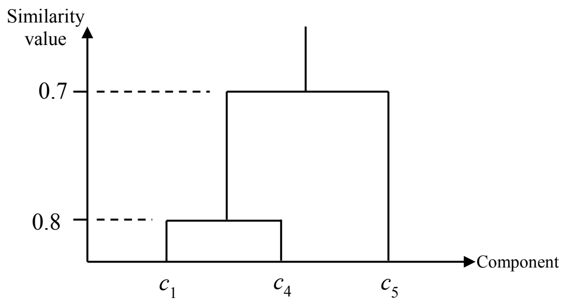
4.4.2. Tree construction

The construction of a tree (dendrogram) is a distinctive feature of hierarchical clustering to analyze a set of similarity values between any two objects, and the average linkage method is adapted in this paper (Everitt et al., 2011). To keep the paper self-contained, the basic procedure to construct a tree based on the similarity matrix of components (SC) is provided as follows.

Step 1: find the two components that have the highest similarity value recorded in SC and form a new branch accordingly. For example, c_1 and c_4 have the highest similarity value in Figure 4-2b and therefore they are joined to form a branch in Figure 4-3a.

Step 2: update the similarity matrix based on the new branch. Let $c_{(i,j)}$ denotes the grouped components after joining c_i and c_j as a new branch (e.g. $c_{(1,4)}$ in the previous step). The update process is based on the similarity values of c_i and c_j with other remaining components in SC. Based on the average linkage method, Equation (2) shows the average calculation between the remaining components c_k and newly joined branch of c_i and c_j. Also, Figure 4-3b shows the updated SC after joining c_1 and c_2 in Figure 4-3a.

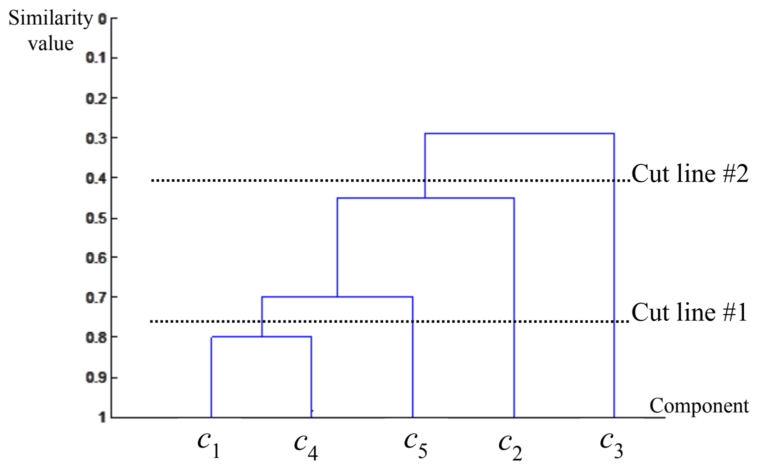
$$sc_{(ij)k} = \frac{sc_{ik} + sc_{jk}}{2} \tag{2}$$



***4-3a** Tree construction based on similarity values*

	$c_{(1,4)}$	c_2	c_3	c_5
$c_{(1,4)}$	-	0.50	0.45	0.70
c_2	0.50	-	0.25	0.40
c_3	0.45	0.25	-	0.20
c_5	0.70	0.40	0.20	-

***4-3b** updated similarity matrix SC after joining c_1 and c_4*



***4-3c**. Result of tree construction with cut lines and joint points*

Figure 4-3 Tree construction and updated similarity matrix

Step 3: repeat steps 1 and 2 until the similarity matrix SC cannot be further reduced (i.e. all the components are grouped under a single branch). Figure 4-3a shows the iteration that $c_{(1,4)}$ and c_5 are picked to form a branch as they have the highest similarity values indicated in Figure 4-3b. The complete tree structure of this example has been shown in Figure 4-3c.

4.4.3. Tree-based analysis

In this sub-section, a guideline is suggested to utilize the grouping information from the tree structure for decision making. Consider two cut lines in Figure 4-3c as the example. If the cut line #1 is applied, four branches are cut, indicating one component group (i.e., $\{c_1, c_4\}$) and three individual components (i.e., c_5, c_2 and c_3). Alternately, if the cut line #2 is applied, two branches are cut, indicating one component group (i.e., $\{c_1, c_4, c_5, c_2\}$) and one individual component (i.e., c_3).

In this guideline, the position of cut lines reflects the threshold of similarity values to define a component group (i.e., a vanilla box). If higher threshold for similarity values are used (i.e., cut line #1), more groups with smaller group sizes will be obtained. Alternatively, if lower similarity values are set (i.e., cut line #2), it is expected to receive less number of groups with larger group sizes.

The guideline is only intended to provide some ideas about how to use the tree as the graphical information to finalize the grouping results. From the algorithmic view, the benefit of using tree-based analysis is to narrow down the possible solutions in the grouping process. For instance, even for the small sample example, the tree-based analysis narrow down the grouping by suggesting to group c_1 and c_4 first rather than other possibilities (e.g. c_1 and c_5).

In the next section, two examples are employed from literature to illustrate and verify the method's utility to tackle VBC problems.

4.5. Numerical Examples

4.5.1. Vanilla box for drill design

This example is adopted from Ben-Arieh et al. (2009) concerning the design of cordless drills that have five product variants (pv_1, pv_2, pv_3, pv_4 and pv_5) based on a total of 23 components (c_1, c_2 , . . . , c_{23}). The relevant product variety matrix is provided in Figure 4-4.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}
pv_1	1	1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0	1	0	1	0	1	0
pv_2	1	0	1	0	0	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0	1	1	0
pv_3	1	0	0	1	0	1	0	1	1	1	1	0	0	0	0	1	0	0	0	0	1	1	0
pv_4	1	0	0	1	0	1	0	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	1
pv_5	1	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	0	1

Figure 4-4 Product variety matrix of cordless drills

Figure 4-5 shows the tree structure after applying the hierarchical clustering over the components and identifies four major vanilla boxes of components by applying the cut line. Notably, the components of the same box have the highest similarity value (i.e. one), implying that they are used exclusively by some product variants. For example, Box 4 contains $\{c_3, c_{13}, c_{15}, c_{19}\}$ which are only required by pv_2 .

Based on the tree information, five product variants can be described by four vanilla boxes and five individual components. Figure 4-6a shows the corresponding structure concerning the composition of product variants. First of all, Box 1 contains the components required by all product variants and it is the common platform for all product variants. Then, some product variants have their own unique elements (i.e. Box 3 for pv_1, Box 4 for pv_2 , and c_{17} for pv_5), which are shown at the top of the Figure 4-6a. The remaining elements (i.e. Box 2, c_4, c_{16}, c_{22} and c_{23}) are used by product variants in a mixed manner.

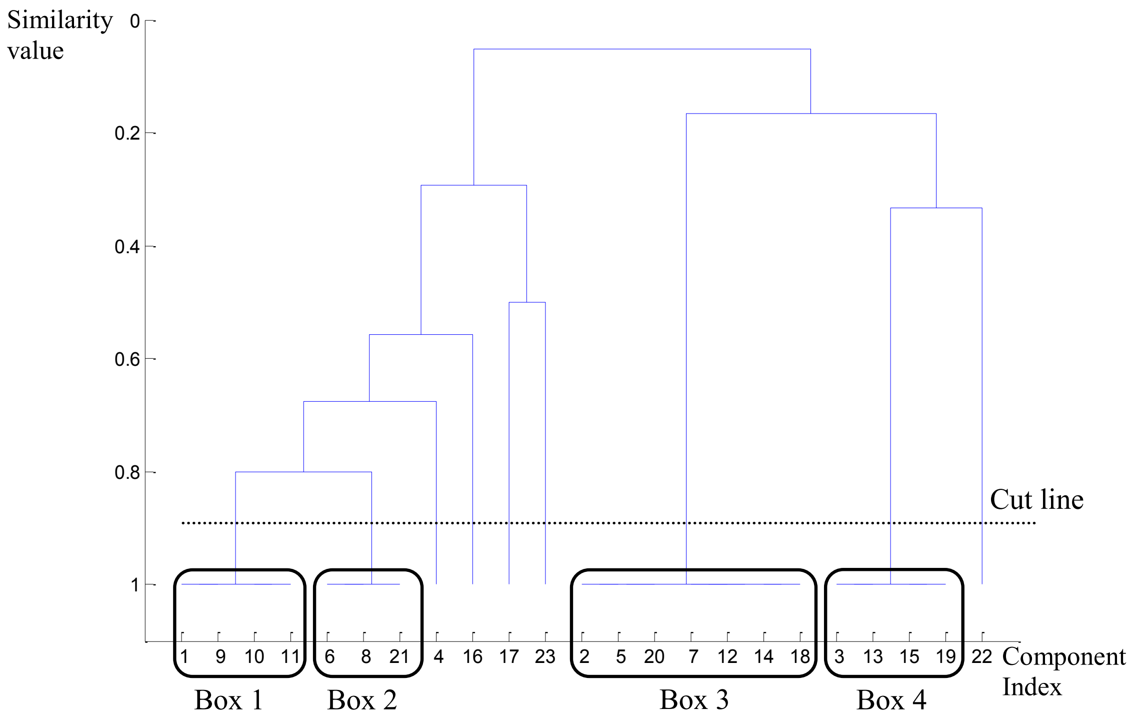
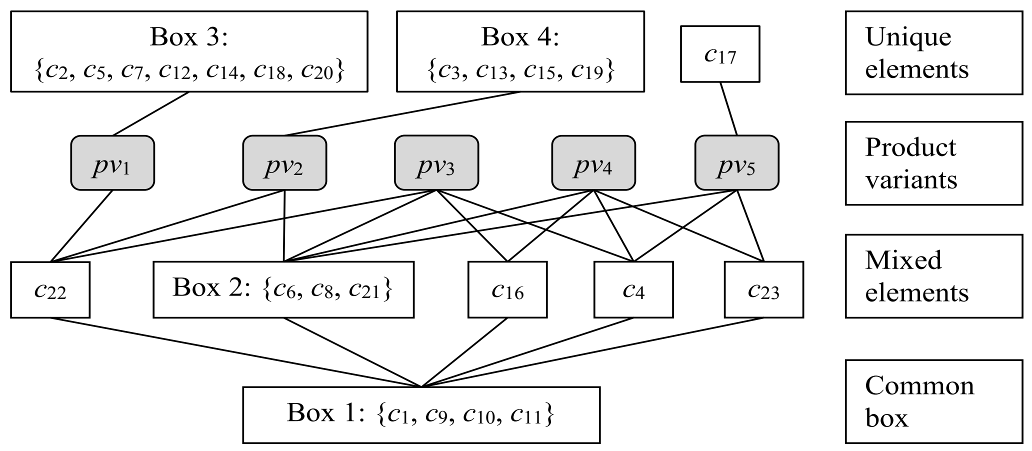


Figure 4-5 Tree structure of cordless drills

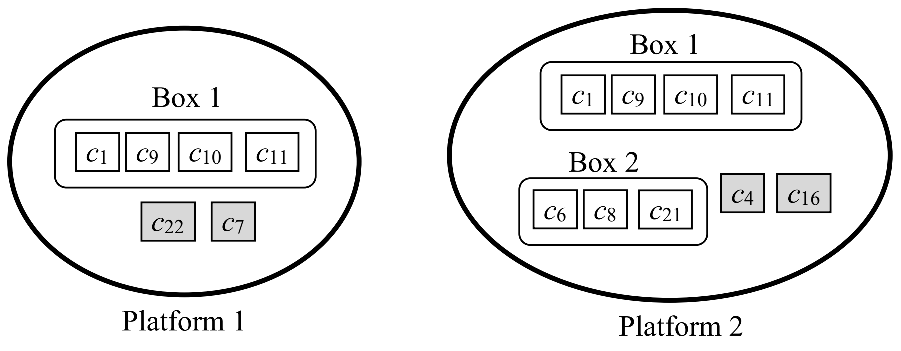
To verify the utility of vanilla boxes in this example, the platform solution from the original paper (Ben-Arieh et al., 2009) are used for comparison purpose. Based on their exact method (using OPL 3.5), two platforms were suggested. Platform 1 consists of six components $\{c_1, c_7, c_9, c_{10}, c_{11}, c_{22}\}$ and it is used for pv_4 and pv_5. Platform 2 consists of nine component $\{c_1, c_4, c_6, c_8, c_9, c_{10}, c_{11}, c_{16}, c_{21}\}$ and it is used for pv_1, pv_2 and pv_3.

The verification effort here is to examine whether the vanilla boxes reasonably correspond to the platform solutions in this original paper. Figure 4-6b shows the components of the platforms in the oval shapes, in which the components of vanilla boxes are grouped in the rounded rectangle. The platform components that do not belong to the vanilla boxes are shaded in grey color.



Box 3: {c_2, c_5, c_7, c_{12}, c_{14}, c_{18}, c_{20}}
Box 4: {c_3, c_{13}, c_{15}, c_{19}}
c_{17}
Unique elements
pv_1
pv_2
pv_3
pv_4
pv_5
Product variants
c_{22}
Box 2: {c_6, c_8, c_{21}}
c_{16}
c_4
c_{23}
Mixed elements
Box 1: {c_1, c_9, c_{10}, c_{11}}
Common box

***4-6a**. Product variety structure for cordless drills*



***4-6b**. Comparison of vanilla boxes and platform solutions*

Figure 4-6 Cordless drills

In the detailed comparison, as Box 1 is required by all the product variants, all of its components are also found in both platforms 1 and 2. Similarly, as Box 2 is required by four product variants, its components are also found in platform 2. This observation supports that the vanilla boxes can be used for determining the platforms.

While the grey shaded components in Figure 4-6b can be viewed as the exceptions, it can be explained by the precedence relationship of assembly operation based on the original paper.

Also, as both Box 3 and Box 4 are unique to pv_1 and pv_2 respectively, it is reasonable that they are not part of the platform solutions.

Notably, the platform solutions in original paper are obtained based on some additional factors (e.g., cost of components) that are not considered in this formation of vanilla boxes. Thus, it is rather reasonable not to expect an exact match between vanilla boxes and platform solutions. Yet, though the information in the product variety matrix is simple, the resulting vanilla boxes are not very different from the platform solutions. The next sub-section will show how to incorporate demand information as an additional factor in the hierarchical process.

4.5.2. Incorporating demand information in VBC problems

4.5.2.1 Background

This example is based on the demonstrative example from Ben-Arieh et al. (2009), and it is chosen because this paper clearly provides details and solutions for comparative study. In this example, four product variants (pv_1, pv_2, pv_3, pv_4) are produced based on eight components (denotes as, *A, B*, *C*, …, *H* according to original symbols). The product variety matrix is provided in Figure 4-7.

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Cost	\$10	\$11	\$12	\$13	\$14	\$15	\$16	\$17
pv_1	1	1	1	1	1	0	0	0
pv_2	1	1	0	1	1	1	0	0
pv_3	1	1	1	0	1	1	0	0
pv_4	1	1	1	0	0	0	1	1

Figure 4-7 Product variety matrix in example 2

Moreover, the demands of product variants are also considered and five scenarios are defined based on original paper. Let d_i denote the demand for ith product variant and the scenarios are provided as follows:

- Scenario 1: [d_1, d_2, d_3, d_4] = [250, 250, 250, 250]
- Scenario 2: [d_1, d_2, d_3, d_4] = [700, 100, 100, 100]
- Scenario 3: [d_1, d_2, d_3, d_4] = [100, 700, 100, 100]
- Scenario 4: [d_1, d_2, d_3, d_4] = [100, 100, 700, 100]
- Scenario 5: [d_1, d_2, d_3, d_4] = [25, 25, 25, 925]

Here, the question is how to configure the vanilla box (or form a platform) in order to reduce the production costs. The purpose of this comparison study is twofold. First, it is intended to observe how the tree structure will change with different demands. Secondly, it is to investigate whether the tree structures correspond to the optimal solutions recorded in the original paper.

4.5.2.2 Adjustment of similarity values based on demands

Compared to the hierarchical clustering process discussed in Section 4.4, the solution process here additionally incorporates the information of demands. In this way, after computing the similarity values based on product variety matrix, the second step adjusts the similarity values based on the demand information. Let r_j denote the demand ratio of the jth component and its formulation is as follows:

$$r_j = \sum_{i=1}^{m} m_{ij} \times \frac{d_i}{\sum d_i} \tag{3}$$

For example, the demand ratio of component C in Scenario 2 is equal to $r_3 = 0.7 + 0 + 0.1 + 0.1 = 0.9$. Here, the demand ratio corresponds to the number of components required in a scenario. Therefore, as Scenario 2 has a total demand of 1000 units, the required number of component C is equal to $0.9 \times 1000 = 900$.

In the similarity adjustment, the basic idea is that components should be grouped in a platform if their demands are high. Let as_{ij} be the adjusted similarity between ith and jth components and its formulation is as follows:

$$as_{ij} = sc_{ij} \times (r_i + r_j) \tag{4}$$

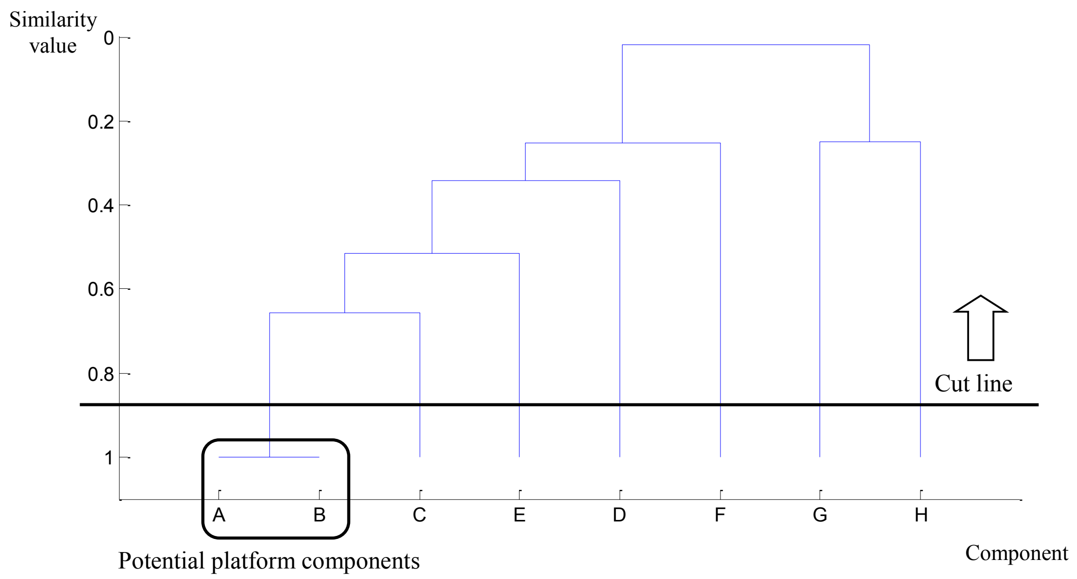
In this formulation, the adjustment factor (i.e. $(r_i + r_j)$) indicates that the similarity value between the ith and jth component is increased according to their demand ratios. As a result, the adjusted similarity matrix $AS = [as_{ij}]$ is obtained to record the adjusted similarity values between components. Since as_{ij} can be larger than one, they will be dividing by the maximum value in the matrix AS for normalization purpose. Then, the normalized AS will become the input for tree construction. The next sub-section discusses the results based on the five scenarios specified earlier.

4.5.2.3 Results and discussion

To examine the quality of the results, the cost data is adapted from the original paper which includes cost of components ($T_c = \{t_{cp} + t_{ca} + t_{cr}\}$), cost of assembling components in a platform (T_p= \$2 per component), cost of adding components (T_a= \$4 per component) or removing components (T_r= \$3 per component) from a platform. Also the platform setup cost (A) is \$1000 per platform. Then the cost formulation for producing pv_i with demand rate of d_i is:

$$T_i = \left(\sum_{c \in C} (T_p + t_{cp}) \times d_i + \sum_{c \in C} \left(T_a + t_{ca}\right) \times d_i + \sum_{c \in C} (T_r - t_{cr}) \times d_i \right) + A \tag{5}$$

In this formulation, T_i is the total cost for producing pv_i; t_{cp} is the cost of component included in the platform; t_{ca} is the cost of component adding to the platform and t_{cr} is the cost of component that is removed from the platform for producing pv_i.



***4-8a**. Potential platform components*

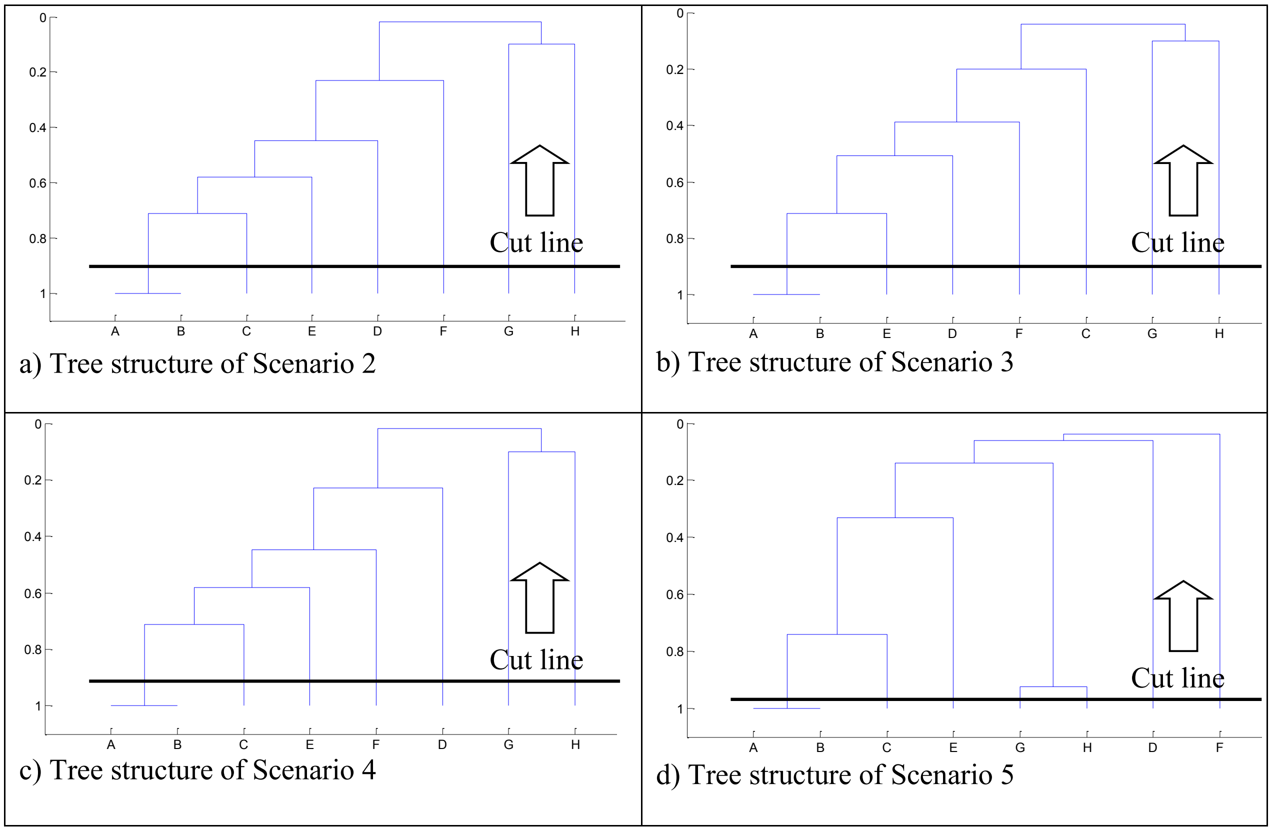
Tree-based platform solution	Cost
{*A*,*B*}	$79750
{*A*,*B*,*C*}	$81750
{*A*,*B*,*C*,*E*}	$84250

***4-8b**. Platform solutions and their costs*

Figure 4-8 Tree structure and platform solutions for Scenario 1

The tree structure for Scenario 1 is presented in Figure 4-8a. By moving the cut line of the tree upward, more components are added to the platform progressively. As low similarity values imply weak tendency to place components in a same group, it is generally suggested not to move the cut line lower than the similarity value of 0.5. Figure 8b lists three platform solutions based on the tree information accordingly. As seen in Figure 4-8b, the platform with lower cost is $\{A, B\}$, and it matches the one-platform result in the original paper.

Figure 4-9 shows the tree structures for Scenario 2 to 5. As components A and B are the common components for all product variants, they are always involved in a platform if the formation of the platform is justified (e.g., low setup cost). In Scenarios 2 to 4, components G and H are grouped on one side and other components on the other side. Then, the tree-based solution for these tree scenarios are synthesized by moving the cut line upward and adding components to $\{A, B\}$ progressively.



a) Tree structure of Scenario 2

b) Tree structure of Scenario 3

c) Tree structure of Scenario 4

d) Tree structure of Scenario 5

Figure 4-9 Tree structure of Scenarios 2 to 5

Table 4-1 lists the tree-based solutions by moving the cut line and their costs based on the original data. As seen, the tree-based solution of the lowest costs in Scenarios 2 and 4 match the platform solution in the original paper. In Scenario 3, the platform of the lowest cost is $\{A, B, E\}$, but it is not a feasible solution due to assembly constraints. Therefore, the original paper identified platform $\{A, B\}$ as the solution platform. As assembly constraints are not considered in forming vanilla boxes, it can be considered that the tree based solution is showing the platform in Scenario 3 with lowest cost.

Table 4-1 platform solutions and their costs in Scenarios 2, 3, 4 and 5

Scenario 2		**Scenario 3**		**Scenario 4**	
Tree-based solution	Cost	Tree-based solution	Cost	Tree-based solution	Cost
{A,B}	$78100	{A,B}	$79900	{A,B}	$79300
{A,B,C}	$77700	{A,B,E}	$79700	{A,B,C}	$78900
{A,B,C,E}	$77500	{A,B,E,D}	$81100	{A,B,C,E}	$78700
{A,B,C,E,D }	$78900	{A,B,E,D,F}	$82900	{A,B,C,E,F}	$80500
Original solution: {A,B,C,E}		Original solution: {A,B}		Original solution: {A,B,C,E}	

Scenario 5				
One-platform solution		**Two-platform solution**		
Platform	Cost	Platform for pv_1, pv_2, pv_3	Platform for pv_4	Cost
{A, B}	$82675	{A,B}	{A,B,G,H}	$79975
{A,B,G,H}	$81750	**{A,B}**	**{A,B,C,G,H}**	**$78125**
{A,B,C,G,H}	**$80150**	{A,B,C}	{A,B,C,G,H}	$78375
{A,B,C,E,G,H}	$94800	{A,B,C,E}	{A,B,C,G,H}	$78225

In Scenario 5, one special feature of the tree structure is that components G and H are joined at a high similarity value. This feature can be explained by the high demand of pv_4, which requires G and H exclusively. This observation verifies that the tree structure is sensitive to the demand information. Based on this specific tree structure, the solutions of one and two platforms can be generated based on the following ideas.

- One-platform solution: starting from $\{A, B\}$, the cut line is moved upward to include the components for forming the platform progressively.
- Two-platform solution: one platform is designated for pv_4 due to its high demand. Then, another platform is made for pv_1, pv_2 and pv_3. While the component set $\{A, B\}$ is

common for both platforms, $\{G,H\}$ is set for pv_4 exclusively. Then, components C and E are used to set different platforms.

The costs of these platforms are listed in Table 1. Notably, in Scenario 5 both one-platform and two-platform solutions with the lowest costs (shaded in Table 1) match the optimal solutions indicated in the original paper.

Computationally, the possible number of platform solutions in this example is large. Though the best platform solutions are not "immediately" identified from the trees, the tree information can effectively narrow down the potential solutions. It is observed that the tree structure can reasonably indicate which components are good candidates for forming a platform. Yet, the automated solution process for the best solutions often requires more contextual details, and it is deferred to future work.

4.5.3. Remarks on the method's utility

In this section, two examples from Ben-Arieh et al. (2009), are used to demonstrate and verify the proposed method. Compared to the approaches using integer programming and genetic algorithm in Ben-Arieh et al. (2009), the proposed method of this paper is relatively simple, and yet, the solution quality is at least not below par. Besides, the tree structures can provide some comprehensive view for determining the final solutions, while the typical optimization approach usually gives a "point" solution. For example, Table 1 shows one "infeasible" solution $\{A,B,E\}$ in Scenario 3, which has a lower cost. This information may help engineers to explore other possibility to lower the cost (e.g., why $\{A,B,E\}$ is infeasible), while this insight cannot be easily obtained just by a single optimal solution point.

4.6. Conclusion

In support of product variety, this paper has addressed the problem of vanilla box configuration using hierarchical clustering. The key issue is how to group the components in the forms of vanilla boxes for facilitating mass customization through cost reduction. In the solution approach, the basic information of product variety is specified in a matrix format, capturing the relations between product variants and required components. Then, hierarchical clustering is applied over components with further consideration of the demands of product variants. Finally, two numerical examples have been conducted to show that the paper's solutions are comparable to the results in literature while it has advantages in terms of simplifying the process and providing flexibility on considering new factors in the hierarchical process.

Future work in this area includes investigating the integration of optimization and cluster analysis for addressing product variety problems. It is expected to develop flexible strategies for studying various aspects of product variety such as design, manufacturing and supply chain in an integrated manner.

4.7. Author's Notes and Significance of Paper to Thesis

This chapter focuses on managing product variety at manufacturing level. The platform development problems in case of product variety is investigated in this chapter. Hierarchical clustering is applied over product components to identify platform configuration in order to reduce production cost. Two example from the literature have been presented and the platform results of the proposed method based on hierarchical clustering are compared with the results of platform results proposed by optimization techniques. It is demonstrated that the proposed method of this paper can yield a good results according to the cost performance metric.

In summary, in this research hierarchical clustering is employed to solve variety-related problems in different stages of product lifecycle including design, manufacturing and supply chain. It's demonstrated that hierarchical clustering is a simple yet reliable approach in comparison with meta-heuristic approaches like genetic algorithms for managing product variety. Moreover, this chapter presents how to consider additional decision factors in the configuration of product platforms. This shows the flexibility of the proposed approach to consider new factor in the structuring process without violating the solution process.

In the next chapter, summary, conclusions, contributions and suggestion for future works are presented.

5. Conclusions

5.1. Summary

In supporting product variety, this thesis has addressed problems of identifying product architecture, platform development and constructing supply chain network using hierarchical clustering. In this way, the key issue was how to bundle the elements in the structuring process considering some performance metrics (e.g. description length, complexity and cost).

In Chapter 2, DSM has been employed to model product architecting problem. A clustering method is developed to identify composition of modules in terms of different sets of components as well as the key interfaces between them. The methodical approach is based on matrix-based hierarchical clustering, and it consists of three phases: (1) components filtering, (2) approximate structure formation, and (3) partitioning analysis. Minimum description length (MDL) objective function has been served as measure to compare the results with the results of the clustering arrangements proposed by GA and also human experts. The results reveal that the proposed clustering method is capable to identify product architecture in favor of product variety management while offering comparable results in view of existing solutions by manual clustering and GA approaches.

In Chapter 3, a methodical procedure is proposed for constructing supply chain network. The proposed method is based on hierarchical clustering, and the core technique is to analyze the coupling values between the product's modules according to the product variety information such as product variants and mix ratios. Five illustrative examples have been developed, and the solutions of constructing supply chain network have been enumerated exhaustively to support that the proposed method can yield a good quality solution. Moreover, the application of the proposed method in terms of managing the postponement strategy has been discussed. It is demonstrated that the proposed method is capable to identify and postpone differentiating modules to downstream processes in supply chain.

In Chapter 4, the problem of vanilla box configuration has been addressed. The key issue is how to group the components in forms of vanilla boxes for facilitating production process in terms of reducing setup cost/time. In the solution approach, the basic information of product variety is specified in a matrix format, capturing the relations between product variants and required components. Then hierarchical clustering is applied over components by further considering some additional factors like demand. Finally two numerical example have been conducted to show that the paper's solutions can provide good solutions in comparison with the results in literature.

5.2. Concluding Remarks on the Utility of Hierarchical Clustering

In this research hierarchical clustering has been employed for tackling product variety problems that are usually solved by optimization techniques due to their combinatorial complexity. Table 5-1 compares the key characteristics of the proposed method in terms of solution approach and obtained results with the existing approaches reported in the literature.

Moreover, in this section, the utility of the proposed solution approach for solving product variety problems is verified based on three aspects including optimality of the results, simplicity of the solution approach and flexibility of the method.

Table 5-1 Comparison of the proposed solution method with existing approaches

Problem Nature	Solution Approach		Results	
	Proposed Method	Existing Approach	Proposed Method	Existing Approach
Paper 1: Formation of modules	➢ Filtering bus components ➢ Similarity of components based on their interactions ➢ Greedy algorithm for grouping similar components	➢ Binary variables assigned to pre-defined number of modules ➢ Meta-heuristic approach to find structure with lower MDL	➢ Better identification of bus components ➢ Poorer f_2 (Zero inside) ➢ Better f_3 (Non-zero outside) ➢ Computational time (roughly 30 sec. for each example)	➢ Some bus components are identified as interactive components ➢ Better f_2 (Zero inside) ➢ Poorer f_3 (Non-zero outside) ➢ Computational time not reported
Paper 2: Formation of sub-assemblies	➢ Similarity of module options based on their usage in final product variants ➢ Incorporating mix ratio information to adjust the coupling values	➢ Take the exhaustive search approach over a small example (a product with 4 modules)	➢ Find optimal solutions in two examples with 4 modules ➢ Find the supply chain structure in an example with 9 modules and 18 module options ➢ Computational time (roughly 20 sec. for each example)	➢ Only done on a small example ➢ No non-exhaustive approach exists for solving larger problems
Paper 3: Formation of semi-finished products	➢ Similarity between components based on their usage in different product variants ➢ Incorporating demand information to adjust the coupling values ➢ Tree analysis to configure vanilla boxes	➢ Binary variables to assign components to the platforms ➢ Meta-heuristic approach to find platform with lower costs	➢ Matching the optimal solution in an example with 23 components ➢ Find vanilla boxes with lower costs in an example with 8 components in 5 different scenarios ➢ Computational time for both examples (roughly 45 sec.)	➢ Find optimal platform configuration ➢ Computational time for an example with 8 components with uniform demands using OPL 3.5 (132 sec.) ➢ Computational time for an example with 23 components with uniform demands using GA (36550 sec.)

In terms of optimality, in designing product architecture problem, the solution by the proposed method is not poorer than other solutions proposed by GA (Yu et al., 2007) and Human expert. Technically, none of the DSM solutions is dominated in view of Pareto optimality. Arguably, it is observed that the proposed method even offers better results in view of product variety management as its solution has lower interfaces between product modules. In terms of configuration of pre-assembled vanilla boxes, compared to the approaches using integer programming and genetic algorithm (Ben-Arieh et al., 2009), the solution quality of the proposed method is at least not below par. Also, in terms of structuring supply chain network, it is observed that the proposed method can suggest a supply chain networks with lower complexity value.

Regarding simplicity of the solution approach, the proposed method offers simpler solutions in terms of computational times and problem formulation. That is, since the proposed approach is based on similarity analysis between any two elements of the product (e.g., components in case of modular design and vanilla box configuration and modules in case of supply chain network), the computational efforts is quite predictable. Alternatively, meta-heuristics techniques require to search over a huge solution space that makes the computational time quite challenging in practice. Also, most optimization problems involve integer (binary) decision variables (Ben-Arieh et al., 2009, Zhang et al., 2013, Kuthambalayan et al., 2014) that causes the complexity of problem formulations. That is, using the "traditional" integer programming algorithms for solving a platform configuration problem, about 12 constraints should be defined to formulate the optimization problem and also some other constraints in terms of "cuts" are required for reducing the size of solution space to make it solvable in terms of computational time (Ben-Arieh et al., 2009). This can lead to long execution times of 36550 seconds for solving a platform configuration problem with 23 component with uniform demand (Ben-Arieh et al., 2009). Comparably, the average computational times for calculating the similarity values and developing the tree for the same problem is about thirty seconds using MATLAB.

In terms of flexibility, the proposed method is quite flexible in terms of adding new factor in the solution approach. That is, considering a new factor (e.g. costs or demand) in case of VBC problem and constructing supply chain network problem is a matter of adjusting coupling values and is not affect other steps of the algorithms. Comparatively, in meta-heuristic approaches like GA, for considering a new factor into consideration, some of the program parameters should be re-tuned according to the new factor that is a challenging task.

In conclusion, the proposed approach based on hierarchical cluster analysis can be a practical tool for managers who need to make decision on product variety issues. It can produce good results in a limited time that at least not below par other methods. Moreover, the proposed method of this paper is relatively simple that makes it easy to implement. It helps to enhance variety management practices through considering similarity between elements while making decision regarding different aspects of product variety management from modular product design to supply chain network and developing vanilla boxes for facilitating the production process.

5.3. Contributions

The original contributions of this thesis concern with both modeling the product variety problems in matrix format and proposing a new solution approach based on hierarchical clustering for solving product variety-related problems that is summarized in Table 5-1. The highlights of these contributions are reported as follows.

- This work has developed a new matrix-based clustering method to identify modular structure of product considering product variety requirements. The proposed clustering algorithm treats explicitly the notion of bus, interactive and module components as the base for developing methodical procedure in order to support offering product variety.
- In terms of supply chain network, this work proposes a new method to construct the structure of supply chain network based on product variety information. The basic information of product variety is specified in the matrix format, capturing the relations

between product variants and required modules. Then, hierarchical clustering is applied over modules by further considering some contextual information such as demand. It proposes a new method for reducing complexity of supply chain network by considering postponement technique.

- This work proposes a new method for tackling platform development problems based on the information of product variants and their components. Two case studies form literature have been conducted to show that the proposed solution approach yields good results in comparison to the results in the literature.

Three journal publications and one conference paper (Appendix A) were used in the body of the entire thesis. In addition, the following publications have been accomplished during the course of the current work:

Journal publications

1. **Pooya Daie** & Simon Li "Hierarchical Clustering for Structuring Supply Chain Network in Case of Product Variety", *Journal of Manufacturing Systems*, Accepted on June 2015 (pending minor revision).

2. **Pooya Daie** & Simon Li "Managing Product Variety through Configuration of Pre-assembled Vanilla Boxes Using Hierarchical Clustering", *International Journal of Production Research*, Submitted on May 2015

3. **Pooya Daie** & Simon Li "Developing Modular Architectures in Support of Product Variety", *Concurrent Engineering, Research & Application*, Submitted on August 2015

Conference publications

1. **Pooya Daie** & Simon Li. 2014. "A Matrix-based Clustering Approach for Developing Modular Architectures", *IIE Conference*, Montreal, QC, Canada.

2. Simon Li & **Pooya Daie**. 2014. "Configuration of Assembly Supply Chain Using Hierarchical Cluster Analysis", *Procedia CIRP* 17: 622-27, Windsor, ON, Canada.

3. **Pooya Daie** & Simon Li. 2012. "Design for Supply Chain", PhD research proposal. *Product Life Cycle Management Conference*, Montreal, QC, Canada.

5.4. Suggestion for Future Research

The possible future directions of this study are highlighted below.

- In terms of DSM clustering, further research is required to find a specific objective function that is aligned with the requirement of product variety in developing modular architectures. The MDL objective function that is used for benchmarking DSM arrangements is not a representative measure as it is designed for benchmarking clustering arrangement and does not consider specific structural requirements of product variety in terms of module interfaces.
- Future works can be done for investigating the integration of optimization and hierarchical clustering in order to address product variety problems. Particularly, hierarchical clustering can be used to narrow down the solution space of some structural problems. Then, optimization techniques can be applied to determine the parametric values to optimize some performance objectives. It is expected to develop flexible strategies for studying various aspects of product variety.

- Future works are needed to broaden the application of hierarchical clustering for addressing product variety problems in some structuring problems. For example, hierarchical clustering can be applied for solving supply chain procurement problem considering effects of components commonality.

References

AGARD, B. & BASSETTO, S. 2013. Modular design of product families for quality and cost. *International Journal of Production Research,* 51**,** 1648-1667.

AL-SALIM, B. & CHOOBINEH, F. 2009. Redesign of production flow lines to postpone product differentiation. *International Journal of Production Research,* 47**,** 5563-5590.

ALEXANDER, C. 1964. *Notes on the Synthesis of Form*, Harvard University Press.

ALGEDDAWY, T. & ELMARAGHY, H. 2012. Product variety management in design and manufacturing: challenges and strategies. *Enabling Manufacturing Competitiveness and Economic Sustainability.* Springer Berlin, 518-523.

BALDWIN, D. F., ABELL, T. E., LUI, M., DE FAZIO, T. & WHITNEY, D. E. 1991. An integrated computer aid for generating and evaluating assembly sequences for mechanical products. *IEEE Transactions on Robotics and Automation,* 7**,** 78-94.

BEN-ARIEH, D., EASTON, T. & CHOUBEY, A. 2009. Solving the multiple platforms configuration problem. *International Journal of Production Research,* 47**,** 1969-1988.

BOAS, R., CAMERON, B. G. & CRAWLEY, E. F. 2013. Divergence and lifecycle offsets in product families with commonality. *Systems Engineering,* 16**,** 175-192.

BROWNING, T. R. 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management* 48**,** 292-306.

BUCKLIN, L. P. 1965. Postponement, speculation and the structure of distribution channels. *Journal of marketing research,* 2**,** 26-31.

CAMERON, B. G. 2011. *Costing commonality: evaluating the impact of platform divergence on internal investment returns.* PhD. Dissertation, Engineering Systems Division, Massachusetts Institute of Technology.

CHENG, Q., ZHANG, G., GU, P. & SHAO, X. 2012. A product module identification approach based on axiomatic design and design structure matrix. *Concurrent Engineering: Research and Applications,* 20**,** 185-194.

DA CUNHA REIS, A., SCAVARDA, L. F. & PANCIERI, B. M. 2013. Product variety management: A synthesis of existing research. *African Journal of Business Management,* 7**,** 39-55.

DA SILVEIRA, G., BORENSTEIN, D. & FOGLIATTO, F. S. 2001. Mass customization: Literature review and research directions. *International journal of production economics,* 72**,** 1-13.

DAVIS, S. M. 1989. From "future perfect": Mass customizing. *Planning review,* 17**,** 16-21.

DAWANDE, M., KESKINOCAK, P., SWAMINATHAN, J. M. & TAYUR, S. 2001. On bipartite and multipartite clique problems. *Journal of Algorithms,* 41**,** 388-403.

ELMARAGHY, H., SCHUH, G., ELMARAGHY, W., PILLER, F., SCHÖNSLEBEN, P., TSENG, M. & BERNARD, A. 2013. Product variety management. *CIRP Annals-Manufacturing Technology,* 62**,** 629-652.

EPPINGER, S. D. & BROWNING, T. R. 2012. *Design structure matrix methods and applications*, MIT press.

EVERITT, B., LANDAU, S., LEESE, M. & STAHL, D. 2011. *Cluster Analysis, Wiley Series in Probability and Statistics*, Chichester, UK: Wiley.

FEITZINGER, E. & LEE, H. L. 1997. Mass Customization at Hewlett-Packard: The power of postponement. *Harvard Business Review,* 75**,** 116-121.

FERNANDEZ, C. 1998. Integration analysis of product architecture to support effective team co-location. *ME thesis, MIT, Cambridge, MA*.

FINE, C. H., GOLANY, B. & NASERALDIN, H. 2005. Modeling tradeoffs in three-dimensional concurrent engineering: a goal programming approach. *Journal of Operations Management,* 23**,** 389-403.

FISHER, M. L. 1997. What is the right supply chain for your product? *Harvard business review,* 75**,** 105-117.

GEN, M. & CHENG, R. 2000. *Genetic algorithms and engineering optimization,* New York, Wiley.

GERSHENSON, J. K., PRASAD, G. J. & ZHANG, Y. 2004. Product modularity: measures and design methods. *Journal of Engineering Design,* 15**,** 33-51.

GRAVES, S. C. & WILLEMS, S. P. 2005. Optimizing the supply chain configuration for new products. *Management Science,* 51**,** 1165-1180.

HALMAN, J. I., HOFER, A. P. & VAN VUUREN, W. 2003. Platform-driven development of product families: linking theory with practice. *Journal of Product Innovation Management,* 20**,** 149-162.

HÖLTTÄ-OTTO, K., TANG, V. & OTTO, K. 2008. Analyzing module commonality for platform design using dendrograms. *Research in Engineering Design,* 19**,** 127-141.

HOMEM DE MELLO, L. & SANDERSON, A. 1991. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation,* 7**,** 228-240.

HU, S., ZHU, X., WANG, H. & KOREN, Y. 2008. Product variety and manufacturing complexity in assembly systems and supply chains. *CIRP Annals-Manufacturing Technology,* 57**,** 45-48.

HU, S. J., KO, J., WEYAND, L., ELMARAGHY, H., LIEN, T., KOREN, Y., BLEY, H., CHRYSSOLOURIS, G., NASR, N. & SHPITALNI, M. 2011. Assembly system design and operations for product variety. *CIRP Annals-Manufacturing Technology,* 60**,** 715-733.

HUANG, G. Q., ZHANG, X. & LIANG, L. 2005. Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. *Journal of Operations Management,* 23**,** 267-290.

JIAO, J. R., SIMPSON, T. W. & SIDDIQUE, Z. 2007. Product family design and platform-based product development: a state-of-the-art review. *Journal of Intelligent Manufacturing,* 18**,** 5-29.

JOHNSON, P. 2003. The Challenge of Complexity in Global Manufacturing. Critical Trends in Supply Chain Management. *Supply Chain Practice,* 5**,** 54-67.

KASHKOUSH, M. & ELMARAGHY, H. 2014. Product Family Formation for Reconfigurable Assembly Systems. *Procedia CIRP,* 17**,** 302-307.

KHAJAVIRAD, A. & MICHALEK, J. J. 2008. A decomposed gradient-based approach for generalized platform selection and variant design in product family optimization. *Journal of Mechanical Design,* 130**,** 071101.

KIM, S.-H. 2014. Postponement for Designing Mass-Customized Supply Chains: Categorization and Framework for Strategic Decision Making. *International Journal of Supply Chain Management,* 3**,** 1-11.

KISPERSKA-MORON, D. & SWIERCZEK, A. 2011. The selected determinants of manufacturing postponement within supply chain context: An international study. *International Journal of Production Economics,* 133**,** 192-200.

KO, J., NAZARIAN, E., WANG, H. & ABELL, J. 2013. An assembly decomposition model for subassembly planning considering imperfect inspection to reduce assembly defect rates. *Journal of Manufacturing Systems,* 32**,** 412-416.

KO, Y.-T. 2013. Optimizing product architecture for complex design. *Concurrent Engineering: Research and Applications,* 21**,** 87-102.

KOREN, Y. & SHPITALNI, M. 2010. Design of reconfigurable manufacturing systems. *Journal of manufacturing systems,* 29**,** 130-141.

KRISHNAN, V. & ULRICH, K. T. 2001. Product development decisions: A review of the literature. *Management science,* 47**,** 1-21.

KUTHAMBALAYAN, T. S., MEHTA, P. & SHANKER, K. 2014. Integrating operations and marketing decisions using delayed differentiation of products and guaranteed delivery time under stochastic demand. *European Journal of Operational Research,* 237**,** 617-627.

LI, S. 2010. Methodical extensions for decomposition of matrix-based design problems. *Journal of Mechanical Design,* 132**,** 061003.

LI, S. 2011. A matrix-based clustering approach for the decomposition of design problems. *Research in Engineering Design,* 22**,** 263-278.

LI, Z., CHENG, Z., FENG, Y. & YANG, J. 2013. An integrated method for flexible platform modular architecture design. *Journal of Engineering Design,* 24**,** 25-44.

LU, C.-C., TSAI, K.-M. & CHEN, J.-H. 2012. Evaluation of manufacturing system redesign with multiple points of product differentiation. *International Journal of Production Research,* 50**,** 7167-7180.

MAN, K.-F., TANG, K.-S. & KWONG, S. 1996. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics,* 43**,** 519-534.

MCAULEY, J. 1972. Machine Grouping for Efficient Production. *Production Engineer,* 51**,** 53-57.

MCCORD, K. R. & EPPINGER, S. D. 1993. *Managing the integration problem in concurrent engineering.* Citeseer.

MIKKOLA, J. H. 2007. Management of product architecture modularity for mass customization: modeling and theoretical considerations. *IEEE Transactions on Engineering Management,* 54**,** 57-69.

NAVAEI, J. & ELMARAGHY, H. 2014. Grouping Product Variants based on Alternate Machines for Each Operation. *Procedia CIRP,* 17**,** 61-66.

NEPAL, B., MONPLAISIR, L. & FAMUYIWA, O. 2012. Matching product architecture with supply chain design. *European Journal of Operational Research,* 216**,** 312-325.

PAGH, J. D. & COOPER, M. C. 1998. Supply chain postponement and speculation strategies: how to choose the right strategy. *Journal of Business Logistics,* 19**,** 13-34.

PARLAKTRK, A. K. & SWAMINATHAN, J. M. 2010. Mass customization. *Wiley Encyclopedia of Operations Research and Management Science*.

PEETERS, R. 2003. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics,* 131**,** 651-654.

PIMMLER, T. U. & EPPINGER, S. D. 1994. Integration analysis of product decompositions. *ASME Design Theory and Methodology Conference.* Minneapolis.

PINE, B. J. & VICTOR, B. 1993. Making mass customization work. *Harvard business review,* 71**,** 108-117.

ROJAS ARCINIEGAS, A. J. & KIM, H. M. 2011. Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric. *Engineering Optimization,* 43**,** 175-192.

SALVADOR, F., FORZA, C. & RUNGTUSANATHAM, M. 2002. Modularity, product variety, production volume, and component sourcing: theorizing beyond generic prescriptions. *Journal of Operations Management,* 20**,** 549-575.

SHARMAN, D. M. & YASSINE, A. A. 2004. Characterizing complex product architectures. *Systems Engineering,* 7**,** 35-60.

SHEKAR, B., VENKATARAM, R. & SATISH, B. 2011. Managing complexity in aircraft design using design structure matrix. *Concurrent Engineering: Research and Applications,* 19**,** 283-294.

SIMPSON, T. W. 2004. Product platform design and customization: Status and promise. *AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 18**,** 3-20.

SIMPSON, T. W., JIAO, J., SIDDIQUE, Z. & HÖLTTÄ-OTTO, K. 2014. Advances in product family and product platform design: Methods & Applications. *New York: Springer*

SIMPSON, T. W., MAIER, J. R. & MISTREE, F. 2001. Product platform design: method and application. *Research in engineering Design,* 13**,** 2-22.

SOSA, M. E., EPPINGER, S. D. & ROWLES, C. M. 2003. Identifying modular and integrative systems and their impact on design team interactions. *Journal of Mechanical Design,* 125**,** 240-252.

SWAMINATHAN, J. M. & TAYUR, S. R. 1998. Managing broader product lines through delayed differentiation using vanilla boxes. *Management Science,* 44**,** 161-172.

ULRICH, K. 1995. The role of product architecture in the manufacturing firm. *Research policy,* 24**,** 419-440.

ULRICH, K. T. & EPPINGER, S. 2012. Product Design and Development. McGraw-Hill, New York, NY.

VENKATESH, S. & SWAMINATHAN, J. M. 2004. Managing product variety through postponement: concept and applications. *The Practice of Supply Chain Management: Where Theory and Application Converge.* Springer, Kluwer, Boston, MA, 139-55.

WANG, H., AYDIN, G. & HU, S. 2009. A complexity model for assembly supply chains in the presence of product variety and its relationship to cost. *under review*.

WANG, H., AYDIN, G. & HU, S. J. An Entropy-based Complexity Model for Assembly Supply Chains and its Relationship to Inventory Cost in the Presence of Product Variety. http://www.osti.gov/eprints/topicpages/documents/record/1017/4750964.html: E-Print Network.

WANG, H., KO, J., ZHU, X. & HU, S. J. 2010. A complexity model for assembly supply chains and its application to configuration design. *Journal of Manufacturing Science and Engineering,* 132**,** 021005.

WANG, H., WANG, H. & HU, S. J. 2013. Utilizing variant differentiation to mitigate manufacturing complexity in mixed-model assembly systems. *Journal of Manufacturing Systems,* 32**,** 731-740.

WANG, H., ZHU, X., WANG, H., HU, S. J., LIN, Z. & CHEN, G. 2011. Multi-objective optimization of product variety and manufacturing complexity in mixed-model assembly systems. *Journal of Manufacturing Systems,* 30**,** 16-27.

WANG, Z., CHEN, L., ZHAO, X. & ZHOU, W. 2014. Modularity in building mass customization capability: The mediating effects of customization knowledge utilization and business process improvement. *Technovation,* 34**,** 678-687.

WEBBINK, R. F. & HU, S. J. 2005. Automated generation of assembly system-design solutions. *IEEE Transactions on Automation Science and Engineering* 2**,** 32-39.

WONG, H., POTTER, A. & NAIM, M. 2011. Evaluation of postponement in the soluble coffee supply chain: A case study. *International Journal of Production Economics,* 131**,** 355-364.

WONG, H., WIKNER, J. & NAIM, M. 2009. Analysis of form postponement based on optimal positioning of the differentiation point and stocking decisions. *International Journal of Production Research,* 47**,** 1201-1224.

YANG, B. & YANG, Y. 2010. Postponement in supply chain risk management: a complexity perspective. *International Journal of Production Research,* 48**,** 1901-1912.

YIN, Y. & YASUDA, K. 2006. Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. *International Journal of Production Economics,* 101**,** 329-352.

YU, T.-L., YASSINE, A. A. & GOLDBERG, D. E. 2007. An information theoretic method for developing modular architectures using genetic algorithms. *Research in Engineering Design,* 18**,** 91-109.

ZHANG, A., LUO, H. & HUANG, G. Q. 2013. A bi-objective model for supply chain design of dispersed manufacturing in China. *International Journal of Production Economics,* 146**,** 48-58.

ZHU, X., HU, S. J., KOREN, Y. & HUANG, N. 2012. A complexity model for sequence planning in mixed-model assembly lines. *Journal of Manufacturing Systems,* 31**,** 121-130.

Appendix A.

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 17 (2014) 622–627

www.elsevier.com/locate/procedia

Variety Management in Manufacturing. Proceedings of the 47th CIRP Conference on Manufacturing Systems

Configuration of assembly supply chain using hierarchical cluster analysis

Simon Li[a,*], Pooya Daie[b]

[a]*University of Calgary, Calgary, Canada*
[b]*Concordia University, Montreal, Canada*

* Corresponding author. Tel.: +1-403-220-5599; fax: +1-403-282-8406. *E-mail address:* simoli@ucalgary.ca

Abstract

The purpose of this paper is to propose a clustering method that configures assembly supply chains. In mass customization, product variety is often characterized by a modular product structure, in which each module may offer several variants to deliver product variants. In this context, assembly supply chain is referred to a network of suppliers who produce and assemble product variants. Given that the proportion of the product variants is specified, the technical problem is how to configure the structure of the assembly supply chain in order to reduce the complexity incurred by product variety. The technical challenge is that the possible number of configurations grows dramatically with the number of product variants. Also, enumerating the possible configurations in a mathematical framework is not a trivial task. Instead of taking an exhaustive approach, the solution approach of this paper is based on hierarchical cluster analysis, in which the tree structure is applied to configure assembly supply chain. The core technique is to investigate the coupling concept in the context of assembly supply chain to characterize the grouping conditions in the structuring process. Few examples are utilized to demonstrate and verify the methodology.

Keywords: assembly supply chain; hierarchical cluster analysis; coupling analysis

1. Introduction

In this paper, product variety is envisioned via product family architecture [1], in which a product is designed in a modular structure, and the variety is achieved by offering several options for each module. A wide range of product variants can then be synthesized by combining different options of each module. Beyond product modularity, Salvador et al. [2] have studied six cases to investigate the relations between product structures and supply chain configurations.

Based on the concept of product family architecture, Wang et al. [3] modeled an assembly supply chain that described how modules were grouped to deliver sub-assemblies for the final assembler. In this context, the configuration of assembly supply chain is influenced by the proportion of product variants to be offered by the final assembler. For instance, in the case of low variety (i.e., few product variants of high proportions), the final assembler may prefer to assemble most of the modules by itself since the difficulty incurred by product variety is not high. Alternatively, in the case of high variety (i.e., many product variants of even proportions), the final assembler may consider to assign the sub-assembly jobs to some suppliers. This approach actually aligns with the postponement concept that the final assembler can focus on the product differentiation at the later production stage [4]. In this context, the research problem of this paper is how to configure the assembly supply chain based on the modular product structure and the proportions of product variants.

The challenges of the configuration problems in manufacturing have been discussed in the review paper by Hu et al. [5]. One fundamental challenge is that the possible number of configurations is numerous even for a small number of elements (e.g., machines or modules). The early relevant works are found in the context of assembly sequencing, including the cut-set method [6] and the algorithm based on the relational model [7]. Webbink and Hu [8] have applied the grouping corollary to generate possible system configurations analytically. By considering the issues of product variety, the complexity measure has been developed to quantify the complexity aspect related to the assembly system supporting product variety [3, 9].

To choose the configurations that can minimize the complexity measures, the basic approach is mainly based on three general steps: (1) generate all possible configurations, (2) compute the complexity measures, and (3) select the configuration of minimum complexity [9]. Some techniques to expedite the solution process include the elimination of the asymmetric configurations [10] and the use of mathematical propositions for guidance [3]. Compared with the configuration efforts in assembly sequences and supply chains, the application of cluster analysis as the solution approach is relatively limited in literature. This paper is intended to contribute in this research direction.

Generally, the complexity issue in the configurations of assembly supply chains stems from the large number of possible configurations. To manage the complexity due to the high number of elements, one intuitive approach is to group the highly-linked elements and identify the structure of the groups. Cluster analysis is basically applied to facilitate the grouping process in the formation of configurations. The specific new concept of this paper is to formulate the coupling values based on the product variety information. Notably, cluster analysis has been recognized as one approach for solving cell formation problems [11].

Nomenclature

$ap_{i,j}$	adjusted coupling value between the *i*th and *j*th module options
br_{ij}	binary relation between the *i*th module option and the *j*th product variant
C	complexity measure
$cp_{i,j}$	coupling value between the *i*th and *j*th module options
e	total number of edges of the assembly supply chain
e_i	number of input edges of the *i*th supplier
md_{i}	*i*th module
$md_{i,j}$	*j*th option of the *i*th module
n	total number of product variants
nq_i	normalized mix ratio of the *i*th module option
p	total number of modules
pv_i	*i*th product variant
$q_{\mathrm{i},v}$	mix ratio of the *i*th supplier for producing the *v*th options or variants
sp_i	*i*th supplier

2. Background: Assembly Supply Chain and Complexity

To maintain the readability of this paper, we briefly discuss the model of the assembly supply chain and the corresponding complexity measure. For readers who want more details, please refer to [3].

In the problem context, the company plans to produce a product that comes with multiple variants. Let $PV = \{pv_1, pv_2, \ldots, pv_n\}$ be the set of *n* product variants. To achieve product variety, all product variants share the same modular product structure. Let $MD = \{md_1, md_2, \ldots, md_p\}$ be the set of p modules. Each module can offer more than one option, and let $md_{i,j}$ denote the jth option of the ith module. Figure 1 illustrates three modules, and each module has two options. Correspondingly, Figure 2 illustrates three possible product variants. Notably, the total possible number of product variants in this case is the multiplication of the numbers of module options (i.e., 2*2*2 = 8).

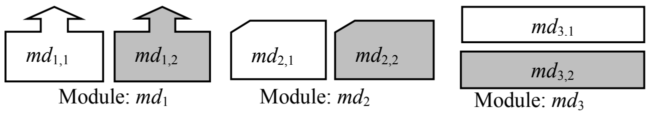


Fig. 1. Illustration of modules and module options.

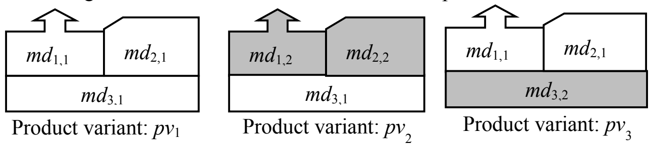


Fig. 2. Illustration of product variants.

With reference to [3], a sample structure of an assembly supply chain is given in Figure 3, in which the node sp_i stands for the ith supplier. There are two special nodes at both ends of the assembly supply chain (i.e., sp_0 and sp_5). The node sp_0 can be viewed as a provider of raw materials, and the node sp_5 is the final assembler who yields the final product variants. Except these two special nodes, other nodes have only one output edge for producing a set of module options or sub-assemblies based on what they receive. To indicate the proportion of different module options or sub-assemblies to be produced from a supplier, a mix ratio will be assigned to each supplier (except sp_0). Let $q_{i,v}$ be the mix ratio of the ith supplier for producing the vth module options, sub-assemblies or product variants. When the mix ratio of the product variants (output of the final assembler) is known, the mix ratios pertaining to the suppliers can be determined. Table 1 lists one example of the mix ratios according to the assembly supply chain demonstrated in Figures 1, 2 and 3.

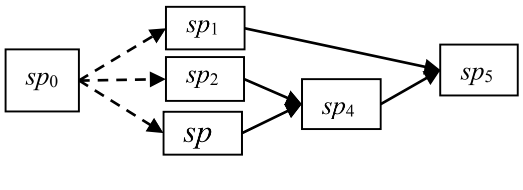


Fig. 3. An example of assembly supply chain.

In assembly supply chain, the complexity measure has been developed based on the concept of information entropy, and the formulation of the complexity measure according to [3] (denoted as C) is given as follow.

$$C = \log_2 e - \frac{1}{e}\sum_i e_i \sum_v q_{i,v} \log_2 q_{i,v} \qquad (1)$$

where e_i is the number of input edges of the ith supplier, and e is the total number of edges calculated by $e = \sum e_i$. Given the mix ratio of product variants, the complexity measure is influenced by the structural configuration (i.e., e and e_i) and the mix ratios of suppliers (i.e., $q_{i,v}$). In brief, more edges and even mix ratios lead to higher complexity. Concerning the modularity of the assembly supply chain [9], the non-modular configuration tends to have less number of edges, and the modular configuration provides some opportunity to produce sub-assemblies of "uneven" mix ratios (i.e., high ratio value on particular variants, leading to lower complexity).

Given the model of assembly supply chain and the complexity measure, the technical question is how to configure the assembly supply chain in order to minimize the complexity measure.

Table 1. Example of the information of each supplier.

Supplier	Output	Mix ratio	No. of input edges
sp_1	$\{md_{1,1}, md_{1,2}\}$	$[q_{1,1}; q_{1,2}] = [0.8; 0.2]$	$e_1 = 1$
sp_2	$\{md_{2,1}, md_{2,2}\}$	$[q_{2,1}; q_{2,2}] = [0.8; 0.2]$	$e_2 = 1$
sp_3	$\{md_{3,1}, md_{3,2}\}$	$[q_{3,1}; q_{3,2}] = [0.9; 0.1]$	$e_3 = 1$
sp_4	$\{(md_{1,1}, md_{2,1}), (md_{1,1}, md_{2,2}), (md_{1,2}, md_{2,1}), (md_{1,2}, md_{2,2})\}$	$[q_{4,1}; q_{4,2}; q_{4,3}; q_{4,4}] = [0.8; 0; 0; 0.2]$	$e_4 = 2$
sp_5	$\{pv_1, pv_2, pv_3\}$	$[q_{5,1}; q_{5,2}; q_{5,3}] = [0.7; 0.2; 0.1]$	$e_5 = 2$

3. Methodology

Concept of coupling and assembly supply chain

The foundation of the methodology is based on hierarchical cluster analysis (HCA). In HCA, one crucial step before the clustering process is to evaluate the similarity (or dissimilarity) between any two objects. The clustering process tends to group the objects that share high similarity values with each other. Since the term "similarity" may imply the common traits between two objects (e.g., both flies and eagles have wings), this paper uses the term "coupling" to imply a more general concept. Particularly, the coupling is referred to the appropriateness of two objects to be grouped in one application. If it is appropriate to group two objects in one application, the corresponding coupling value is high. Notably, the coupling value is application-dependent. That is, we may consider that objects a and *b* should be grouped in one case but not be grouped in another case. Then, the application context becomes the important information to guide the formulation of coupling values.

In assembly supply chain, the key "coupling" question is whether two modules should be grouped together and to explain why. For example, in Figure 3, should we group md_1 and md_2? If so (or not), what is the reason? As any intermediate supplier (to produce sub-assemblies) will incur more edges (i.e., higher complexity), it is important for an intermediate supplier to yield a "low-variety" mix ratio to counter the additional edges. In view of coupling, if two modules can be combined to yield a "low-variety" mix ratio, the coupling values between these two modules will be high. Based on this idea, the detailed quantification of the coupling values is developed in the methodical procedure.

Notably, the procedure for HCA (e.g., building the dendrogram or tree) has been well documented in literature (e.g., [12]), and it will not be repeated here. The methodical procedure in the next sub-section will focus on how to compute the coupling values for building the tree and how to configure the assembly supply chain based on the tree result.

Methodical procedure

Step 1: fill in the product variety form

This step is to collect the input information of product variety and fill it in a compact form, namely, the product variety form. This form consists of two major parts. One part is the binary matrix that captures the relations between module options and product variants. The gray area in Figure 4 shows such a matrix according to the example in Figure 2. Let br_{ij} be the binary matrix entry that $br_{ij} = 1$ if the *i*th module option is selected in the *j*th product variant (else, $br_{ij} = 0$).

Another part is to record the mix ratios of module options and product variants, which are placed along the corresponding rows and columns, respectively. For illustration, the mix ratios of the example based on Table 1 are also recorded in Figure 4. For normalization, the mix ratios of module options are divided by the number of modules (i.e., 3 in the example), and the normalized value of the *i*th module option is denoted as nq_i (i.e., the last column of Figure 4).

	pv_1 (mix ratio)	pv_2 (mix ratio)	pv_3 (mix ratio)	Mix ratio of module options	Normalized mix ratio
$md_{1,1}$	$br_{11} = 1$	$br_{12} = 0$	$br_{13} = 1$	$q_{1,1} = 0.80$	$nq_1 = 0.27$
$md_{1,2}$	$br_{21} = 0$	$br_{22} = 1$	$br_{23} = 0$	$q_{1,2} = 0.20$	$nq_2 = 0.07$
$md_{2,1}$	$br_{31} = 1$	$br_{32} = 0$	$br_{33} = 1$	$q_{2,1} = 0.80$	$nq_3 = 0.27$
$md_{2,2}$	$br_{41} = 0$	$br_{42} = 1$	$br_{43} = 0$	$q_{2,2} = 0.20$	$nq_4 = 0.07$
$md_{3,1}$	$br_{51} = 1$	$br_{52} = 1$	$br_{53} = 0$	$q_{3,1} = 0.90$	$nq_5 = 0.30$
$md_{3,2}$	$br_{61} = 0$	$br_{62} = 0$	$br_{63} = 1$	$q_{3,2} = 0.10$	$nq_6 = 0.03$
	$q_{5,1} = 0.70$	$q_{5,2} = 0.20$	$q_{5,3} = 0.10$		

Fig. 4. Product variety form.

Step 2: determine the coupling values of module options

Two module options are coupled if they are selected in the same product variant(s). In the reasoning, if two specific module options are often selected at the same time in many product variants, we can group them together to form a sub-assembly that can be produced with high volume and low variety. Based on the binary matrix in the product variety form, the Jaccard coefficient is applied to evaluate the coupling values. Let $cp_{i,j}$ be the coupling value between the *i*th and *j*th module options, and its formulation is given below.

$$cp_{i,j} = \frac{\sum_{k=1}^{n} \min(br_{ik}, br_{jk})}{\sum_{k=1}^{n} \max(br_{ik}, br_{jk})} \tag{2}$$

Recall that *n* is the total number of product variants. Notably, the min and max operations can be considered as an alternative way to count the 1-1 and 1-0 matches in the Jaccard coefficient. For illustration, Figure 5 shows the square coupling matrix that records the coupling values between two module options of the example.

Step 3: adjust the coupling values based on mix ratios

If the mix ratios of two module options are high, the grouping of these module options can potentially reduce the complexity (for high production of specific variants). Thus, the coupling values computed in the previous step are adjusted based on the information of mix ratios. Let $ap_{i,j}$ be the adjusted coupling value between the *i*th and *j*th module options, and its formulation is given below. Figure 6 shows the matrix of adjusted coupling values for the example.

$$ap_{i,j} = cp_{i,j}(nq_i + nq_j) \tag{3}$$

	$md_{1,1}$	$md_{1,2}$	$md_{2,1}$	$md_{2,2}$	$md_{3,1}$	$md_{3,2}$
$md_{1,1}$		0	1	0	0.33	0.50
$md_{1,2}$	0		0	1	0.50	0
$md_{2,1}$	1	0		0	0.33	0.50
$md_{2,2}$	0	1	0		0.50	0
$md_{3,1}$	0.33	0.50	0.33	0.50		0
$md_{3,2}$	0.50	0	0.50	0	0	

Fig. 5. Coupling matrix of module options.

	$md_{1,1}$	$md_{1,2}$	$md_{2,1}$	$md_{2,2}$	$md_{3,1}$	$md_{3,2}$
$md_{1,1}$		0	*0.54*	*0*	0.19	0.15
$md_{1,2}$	0		*0*	*0.14*	0.19	0
$md_{2,1}$	0.54	0		0	0.19	0.15
$md_{2,2}$	0	0.14	0		0.19	0
$md_{3,1}$	0.19	0.19	0.19	0.19		0
$md_{3,2}$	0.15	0	0.15	0	0	

Fig. 6. Adjusted coupling matrix.

Step 4: determine the coupling values between modules

When configuring an assembly supply chain, the grouping process is essentially carried out over the modules rather than module options. Thus, this step is to determine the average of the coupling values between two modules by considering all relevant module variants. For example, by checking Figure 6, there are four coupling values between md_1 and md_2 (italicized for highlight), and we can find the average of these four values to reflect the coupling between md_1 and md_2. Figure 7 shows the corresponding results of the example.

	md_1	md_2	md_3
md_1		0.17	0.13
md_2	0.17		0.13
md_3	0.13	0.13	

Fig. 7. Coupling matrix between modules.

Step 5: construct the tree

Based on the coupling matrix between modules obtained from the previous step, the standard procedure of hierarchical cluster analysis is applied by treating coupling values same as similarity measures. The resulting tree of the example is given in Figure 8.

Step 6: cut the tree to suggest the configuration of the assembly supply chain

The top branch of the tree can be considered the position of the final assembler. Then, the tree structure basically suggests how different modules should be grouped leading to the final assembler. Consider two cut lines in Figure 8 as the example. If the cut line #1 is applied, two branches are cut, indicating that the final assembler receives two sub-assemblies, one from md_1 and md_2 and another from md_3. Similarly, if the cut line #2 is applied, three branches are cut, and the non-modular configuration is obtained. The configurations based on these two cut lines are shown in Figure 9, where sp_1, sp_2 and sp_3 produce md_1, md_2 and md_3, respectively.

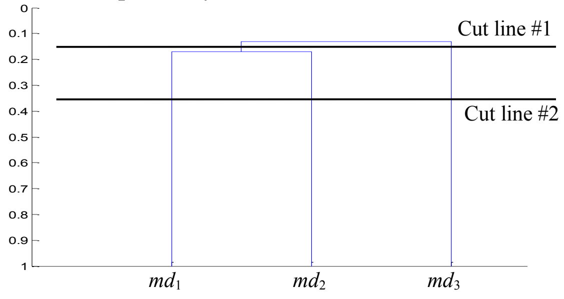


Fig. 8. Tree and cut lines.

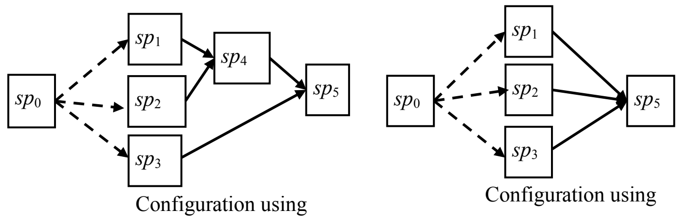


Fig. 9. Configurations and cut lines.

Numerical examples

Five numerical cases are set to examine the proposed method of this paper. The characteristics of these cases are discussed as follows.

- Case #1: this case has three modules to yield two product variants. Two module options are used exclusively.
- Case #2: this case has four modules to yield two product variants. Three module options are used exclusively.
- Case #3: this case has three modules, and each module has two variants. It is required to produce all possible product variants with the same mix ratio.
- Case #4: this case has four modules to yield five product variants. No module option is used exclusively.
- Case #5: this case is similar to Case #4 except that some module options are purposely selected with high mix ratios.

To keep the discussion easy to follow, the numerical details and the results of these cases are organized in the following sub-sections.

Discussion of Cases #1 and #2

The product variety forms of Cases #1 and #2 as the methodical input are provided in Figure 10. While both cases intend to yield two product variants with the same mix ratio (i.e., 0.5), these variants are differentiated only by one module (i.e., md_3 for Case #1 and md_4 for Case #2). Thus, the sensible configurations should group the modules that contribute the common options of all product variants.

The trees and configurations based on the proposed method for Cases #1 and #2 are shown in Figures 11 and 12, respectively. First of all, both trees show that the differentiating modules have lower coupling values with other modules. Following the tree structures, the configurations suggest forming sub-assemblies of the "non-differentiating" modules, and these align with the sensible configurations discussed earlier. Algorithmically, the tree is constructed by grouping two modules at one time. Yet, if three modules have close coupling values with each other, the tree can still yield such a structure approximately, as shown in Case #2.

To compactly represent the configurations, we adapt the "bracket" representation from [3]. For example, the configurations from Figures 11 and 12 can be represented as ((sp_1, sp_2) (sp_3)) and ((sp_1, sp_2, sp_3) (sp_4)). To verify the results, the complexity measure is applied to compare the configurations in Figures 11 and 12 with the non-modular configurations. The complexity measures are provided in Table 2, which shows that the configurations in Figures 11 and 12 have lower complexity.

	pv_1	pv_2	
$md_{1,1}$	1	1	1
$md_{2,1}$	1	1	1
$md_{3,1}$	1	0	0.5
$md_{3,2}$	0	1	0.5
	0.5	0.5	

	pv_1	pv_2	
$md_{1,1}$	1	1	1
$md_{2,1}$	1	1	1
$md_{3,1}$	1	1	1
$md_{4,1}$	1	0	0.5
$md_{4,2}$	0	1	0.5
	0.5	0.5	

Fig. 10. Product variety forms of (a) Case #1 and (b) Case #2.

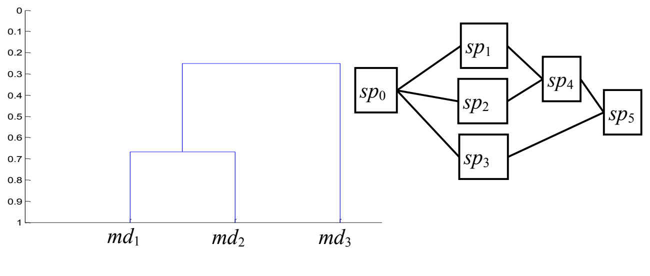


Fig. 11. Results of Case #1 (a) tree and (b) configuration.

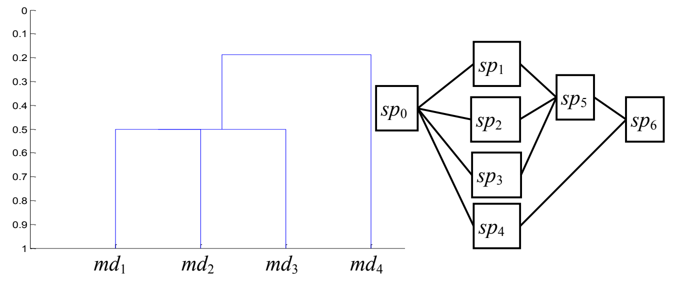


Fig. 12. Results of Case #2(a) tree and (b) configuration.

Table 2. Comparison of Cases #1 and #2 with non-modular configuration.

	Configuration	Complexity
Case #1	((sp_1, sp_2) (sp_3))	3.236
	(sp_1, sp_2, sp_3)	3.252
Case #2	((sp_1, sp_2, sp_3) (sp_4))	3.503
	(sp_1, sp_2, sp_3, sp_4)	3.625

Discussion of Case #3

Different from Cases #1and #2, Case #3 has equal mix ratios of all possible variants that are constructed from three modules and two module options. Figure 13 shows the product variety form of this case, and Figure 14 shows the resulting tree and configuration. Notably, this is a "non-modular" configuration.

According to Proposition 2 in [3], it is generally suggested that "modular" configurations are preferred for the situations of equal demand shares. In their proof of Proposition 2, the complexity of modular configurations is higher than that of non-modular configurations only if the numbers of modules and variants are high enough (to satisfy a threshold condition). In our brief analysis, a modular configuration tends to yield a higher number of edges, which lead to higher complexity measures. Yet, based on the complexity formulation in (1), the increase of $\log_2 e$ is less steep for large e values. This explains why modular configurations are generally preferred in case of higher numbers of modules and variants. However, the proposed method does not discern the situations of higher numbers of modules and variants (from the lower ones) given the equal demand shares of all possible variants. Further research is required to address this special situation.

	pv_1	pv_2	pv_3	pv_4	pv_5	pv_6	pv_7	pv_8	
$md_{1,1}$	1	1	1	1	0	0	0	0	0.5
$md_{1,2}$	0	0	0	0	1	1	1	1	0.5
$md_{2,1}$	1	1	0	0	1	1	0	0	0.5
$md_{2,2}$	0	0	1	1	0	0	1	1	0.5
$md_{3,1}$	1	0	1	0	1	0	1	0	0.5
$md_{3,2}$	0	1	0	1	0	1	0	1	0.5
	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	

Fig. 13. Product variety form of Case #3.

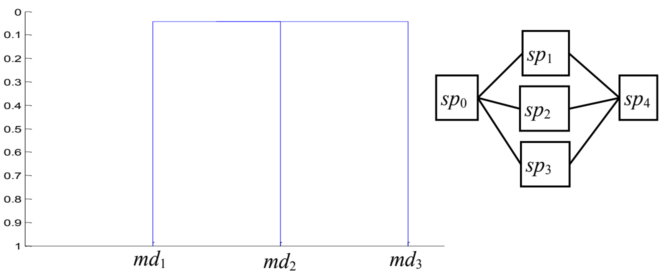


Fig. 14. Results of Case #3 (a) tree and (b) configuration.

	pv_1	pv_2	pv_3	pv_4	pv_5	
$md_{1,1}$	1	1	0	0	0	0.27
$md_{1,2}$	0	0	1	0	0	0.18
$md_{1,3}$	0	0	0	1	1	0.55
$md_{2,1}$	1	0	0	1	1	0.66
$md_{2,3}$	0	1	1	0	0	0.34
$md_{3,1}$	0	0	0	1	0	0.31
$md_{3,2}$	0	1	1	0	0	0.34
$md_{3,3}$	1	0	0	0	1	0.35
$md_{4,1}$	0	1	1	1	0	0.65
$md_{4,2}$	0	0	0	0	1	0.24
$md_{4,3}$	1	0	0	0	0	0.11
	0.11	0.16	0.18	0.31	0.24	

	pv_1	pv_2	pv_3	pv_4	pv_5	
$md_{1,1}$	1	1	1	1	1	1
$md_{2,1}$	1	1	1	1	0	0.89
$md_{2,2}$	0	0	0	0	1	0.11
$md_{3,1}$	1	1	1	0	1	0.69
$md_{3,2}$	0	0	0	1	0	0.31
$md_{4,1}$	1	0	0	1	1	0.66
$md_{4,2}$	0	1	0	0	0	0.16
$md_{4,3}$	0	0	1	0	0	0.18
	0.24	0.16	0.18	0.31	0.11	

Fig. 15. Product variety forms of (a) Case #4 and (b) Case #5.

Discussion of Cases #4 and #5

Both Cases #4 and #5 have four modules, and each module has three options to yield five product variants in total. For investigation purpose, we attempt to "randomly" pick up the module options to construct product variants in Case #4. In contrast, some module options are used more frequent than others in Case #5. The product variety forms of Cases #4 and #5 as the methodical input are provided in Figure 15. The trees and configurations based on the proposed method for Cases #4 and #5 are shown in Figures 16 and 17, respectively.

In Case #4, the highest coupling value that joins two modules (i.e., md_2 and md_3) is about 0.07 (see the tree in Figure 16). If we set the cut line of the tree at 0.1, we will obtain the non-modular configuration that the final assembler addresses all modules without sub-assemblies. In contrast, if the cut line is set at 0.1 in Case #5, we obtain a configuration that groups md_1 & md_2 and then with md_3 (see the tree in Figure 17). This configuration generally makes sense because $md_{1,1}$ and $md_{2,1}$ are used together with high mix ratio 0.89.

To further examine the proposed method, we have identified all possible configurations exhaustively and evaluate the complexity measures for Cases #4 and #5. The results are recorded in Table 3, and the complexity measures of the configurations suggested by the proposed method are highlighted in gray color. As seen in Table 3, the proposed method can suggest the configurations of the lowest complexity for Cases #4 and #5. These results support the utility of the proposed method towards the complexity reduction of an assembly supply chain.

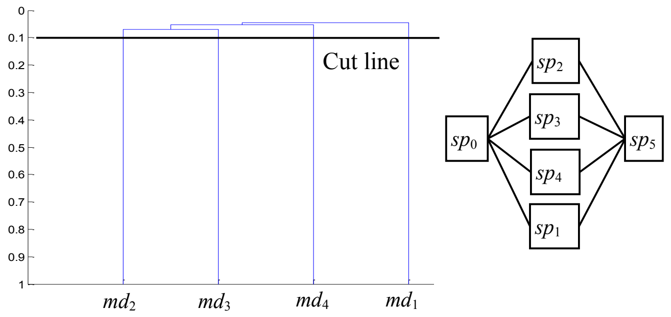


Fig. 16. Results of Case #4 (a) tree and (b) configuration.

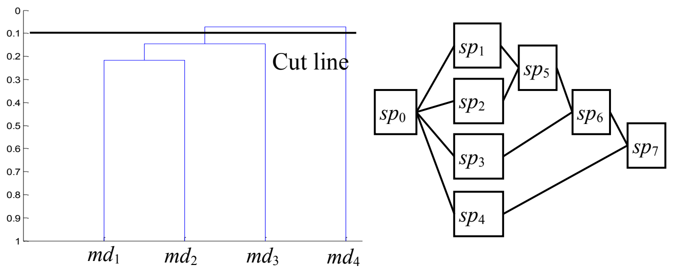


Fig. 17. Results of Case #5 (a) tree and (b) configuration.

Closing Remarks

This paper has proposed a method for configuring the assembly supply chain. The method is based on the adaption of hierarchical cluster analysis, and the core technique is to evaluate the coupling according to the product variety information. The proposed method has been examined by five numerical examples, and the preliminary results have demonstrated the utility of the method in view of suggesting sensible configurations and reducing complexity.

Further research is in progress to improve the proposed method. One direction is to address the special situation that all possible product variants are required with equal demand shares. In this case, the concern is to be sensitive to the absolute number of modules in the configuration process. Another direction is to develop a more detailed guideline to cut the tree and suggest the corresponding configuration.

Table 3. Exhaustive comparison of complexity measures in Cases #4 and #5.

Configuration	Case #4	Case #5
(sp_1, sp_2, sp_3, sp_4)	4.767	4.450
((sp_1, sp_2, sp_3) (sp_4))	4.989	4.405
((sp_2, sp_3, sp_4) (sp_1))	4.876	4.708
((sp_1, sp_3, sp_4) (sp_2))	4.989	4.603
((sp_1, sp_2, sp_4) (sp_3))	4.989	4.526
((sp_1, sp_2) (sp_3, sp_4))	5.006	4.519
((sp_1, sp_3) (sp_2, sp_4))	5.115	4.552
((sp_1, sp_4) (sp_2, sp_3))	5.052	4.554
((sp_1, sp_2) (sp_3) (sp_4))	4.868	4.322
((sp_1, sp_3) (sp_2) (sp_4))	4.989	4.409
((sp_1, sp_4) (sp_2) (sp_3))	4.989	4.492
((sp_2, sp_3) (sp_1) (sp_4))	4.843	4.506
((sp_2, sp_4) (sp_1) (sp_3))	4.913	4.587
((sp_3, sp_4) (sp_1) (sp_2))	4.913	4.638
(((sp_1, sp_2) (sp_3)) (sp_4))	5.074	4.401
(((sp_1, sp_2) (sp_4)) (sp_3))	5.074	4.474
(((sp_1, sp_3) (sp_2)) (sp_4))	5.182	4.480
(((sp_1, sp_3) (sp_4)) (sp_2))	5.182	4.598
(((sp_1, sp_4) (sp_2)) (sp_3))	5.182	4.626
(((sp_1, sp_4) (sp_3)) (sp_2))	5.182	4.672
(((sp_2, sp_3) (sp_1)) (sp_4))	5.052	4.567
(((sp_2, sp_3) (sp_4)) (sp_1))	4.984	4.748
(((sp_2, sp_4) (sp_1)) (sp_3))	5.115	4.712
(((sp_2, sp_4) (sp_3)) (sp_1))	5.047	4.821
(((sp_3, sp_4) (sp_1)) (sp_2))	5.115	4.804
(((sp_3, sp_4) (sp_2)) (sp_1))	5.047	4.867

References

[1] Tseng MM, Jiao J, Merchant ME. Design for mass customization. CIRP Annals – Manufacturing Technology 1996; 45: 153-156.

[2] Salvador F, Rungtusanatham M, Forza C. Supply-chain configurations for mass customization. Production Planning and Control 2004; 15: 381-391.

[3] Wang H, Ko J, Zhu X, Hu SJ. Wang H, Ko J, Zhu X, Hu SJ. A complexity model for assembly supply chains and its application to configuration design. ASME Journal of Manufacturing Science and Engineering 2010; 132: 021005.

[4] Feitzinger E, Lee HL, Mass customization at Hewlett-Packard: the power of postponement. Harvard Business Review 1997; 75: 116-121.

[5] Hu SJ, Ko J, Weyand L, ElMaraghy HA, Lien TK, Koren Y, Bley H, Chryssolouris G, Nasr N, Shpitalni M. Assembly system design and operations for product variety. CIRP Annals – Manufacturing Technology 2011; 60: 715-733.

[6] Baldwin DF, Abell TE, Lui M, De Fazio TL, Whitney DE. An integrated computer aid for generating and evaluating assembly sequences for mechanical products. IEEE Transactions on Robotics and Automation 1991; 7: 78-94.

[7] Homem de Mello LS, Sanderson AC. A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Transactions on Robotics and Automation 1991; 7: 228-240.

[8] Webbink R, Hu SJ. Automatic generation of assembly system solutions. IEEE Transactions on Automation Science and Engineering 2005; 2: 32-39.

[9] Hu SJ, Zhu X, Wang H, Koren Y. Product variety and manufacturing complexity in assembly systems and supply chains, CIRP Annals – Manufacturing Technology 2008; 57: 45-48.

[10] Koren Y, Shpitalni M. Design of reconfigurable manufacturing systems. Journal of Manufacturing Systems 2010; 29: 130-141.

[11] Yin Y, Yasuda K. Similarity coefficient methods applied to the cell formation problem: a taxonomy and review. International Journal of Production Economics 2006; 101: 329-352.

[12] Everitt BS, Landau S, Leese M, Stahl D. Cluster analysis. 5th ed. West Sussex: John Wiley and Sons; 2011.