7-28-2015

# Knowledge Discovery Models for Product Design, Assembly Planning and Manufacturing System Synthesis

Mohamed Kashkoush
*University of Windsor*

# Knowledge Discovery Models for Product Design, Assembly Planning and Manufacturing System Synthesis

By

Mohamed Kashkoush

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Industrial and Manufacturing Systems Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2015

Knowledge Discovery Models for Product Design, Assembly Planning and Manufacturing System Synthesis

by

Mohamed Kashkoush

APPROVED BY:

April 21, 2015

## DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

### I. Co-Authorship Declaration

I hereby declare that this thesis incorporates material that is result of joint research of the author and his supervisor Prof. Hoda ElMaraghy. This joint research has been published / submitted to various Journals that are listed below.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from Prof. Hoda ElMaraghy to include that material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

### II. Declaration of Previous Publication

This thesis includes 9 original papers that have been previously published / submitted for publication in peer reviewed journals and conferences as follows:

| Thesis Chapter | Publication title/full citation | Publication Status |
|---|---|---|
| 2 | Kashkoush, M. and ElMaraghy, H., 2014. *Product Design Retrieval by Matching Bills of Materials*. Journal of Mechanical Design, Transactions of the ASME, Vol. 136(1), pp. 011002-1 - 011002-10. | Journal (Published) |
| 3 | Kashkoush, M. and ElMaraghy, H., 2015. *Product Family Formation by Matching BOM Trees*. Submitted to CIRP Journal of Manufacturing Science and Technology, April 2014, Ref. no.: CIRPJ-D-14-00050. | Journal (Submitted) |
| 4 | Kashkoush, M. and ElMaraghy, H., 2014. *Consensus Tree Method for Generating Master Assembly Sequence*. Production Engineering, Vol. 8, pp. 233–242. | Journal (Published) |
| 4 | Kashkoush, M. and ElMaraghy, H., 2015. *Knowledge-Based Model for Constructing Master Assembly Sequence*. Journal of Manufacturing Systems, Vol. 34, pp. 43-52. | Journal (Published) |
| 5 | Kashkoush, M. and ElMaraghy, H., 2015. *An Integer Programming Model for Discovering Associations between Manufacturing System Capabilities and Product Features*. Journal of Intelligent Manufacturing, DOI: 10.1007/s10845-015-1044-6. | Journal (In Press) |
| 5 | ElMaraghy, H. and Kashkoush, M., 2015. *Assembly System Synthesis Using Association Rule Discovery*. Accepted for publication in The International Journal of Advanced Manufacturing Technology, April 2015, Ref. no.: JAMT-D-15-00351. | Journal (Accepted) |

| | | |
|---|---|---|
| 3 | Kashkoush, M. and ElMaraghy, H., 2014. *Product Family Formation for Reconfigurable Assembly Systems*. In proceeding of the 47th CIRP Conference on Manufacturing Systems (CMS), Windor, ON, Canada, Procedia CIRP, Vol. 17, pp. 302-307. | Conference Proceeding |
| 2 | Kashkoush, M. and ElMaraghy, H., 2013. *Matching Bills of Materials Using Tree Reconciliation*. In proceeding of the 46th CIRP Conference on Manufacturing Systems (CMS), Setubal, Portugal, Procedia CIRP, Vol. 7C, pp. 169-174. | Conference Proceeding |
| 4 | Kashkoush, M. and ElMaraghy, H., 2013. *Generating Master Assembly Sequence Using Consensus Trees*. In proceeding of the 5th Int. Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV), Munich, Germany, pp. 261-266. | Conference Proceeding |

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

# ABSTRACT

The variety of products has been growing over the last few decades so that the challenges for designers and manufacturers to enhance their design and manufacturing capabilities, responsively and cost-effectively are greater than ever. The main objective of this research is to help designers and manufacturers cope with the increasing variety management challenges by exploiting the data records of existing or old products, along with appropriate Knowledge Discovery (KD) models, in order to extract the embedded knowledge in such data and use it to speed-up the development of new products. Four product development activities have been successfully addressed in this research: *product design*, *product family formation*, *assembly sequencing* and *manufacturing system synthesis*. The models and methods developed in this dissertation present a package of knowledge-based solutions that can greatly support product designers and manufacturers at various stages of the product development and manufacturing planning stages.

For design retrieval; using efficient tree reconciliation algorithms found in Biological Sciences, a novel Bill of Materials (BOM) trees matching method was developed to retrieve the closest old design and discover components and structure shared with new product design. As a further application to BOM matching, an enhanced BOM matching method was also developed and used for product family formation. A new approach was introduced for assembly sequencing, based on the notion of consensus trees used in evolutionary studies, to overcome the critical limitation of individual assembly sequence retrieval methods that are not able to capture the assembly sequence data for a given new combination of components that never existed before in the same product variant. For manufacturing system synthesis; a novel Integer Programming model was developed to extract association rules between the product design domain and manufacturing domain to be used for synthesizing a manufacturing/assembly system for new products.

Examples of real products were used to demonstrate and validate the developed models and comparisons with related existing methods were carried out to demonstrate the advantages of the developed models. The outcomes of this research provide efficient, and easy to implement knowledge-based solutions for facilitating cost-effective and rapid product development activities.

## DEDICATION

*To my parents who always push me forward*

*To my wife who always gives me love*

## ACKNOWLEDGEMENT

The author is expressing his great appreciation and gratitude to his supervisor Professor Hoda ElMaraghy for her continuous guidance, support and encouragement. This research would not have been at this level of quality, maturity and completeness without the sincere time and effort she has devoted throughout all its phases. The author also appreciates the valuable feedback from all the committee members that have added to this research. Great thanks should go to Dr. Waguih ElMaraghy, as a committee member and as an Intelligent Manufacturing Systems (IMS) Centre co-director, for his critical suggestions, constructive discussions and valuable comments.

The author acknowledges the help and support he has obtained from the colleagues in the Intelligent Manufacturing Systems (IMS) Centre. Periodic academic meetings as well as continuous informal discussions were a great source for inspiration and feedback; many thanks to Dr. Tarek AlGeddawy, Dr. Mohamed Hanafy, Abdulrahman Seleim, Javad Navaei, Kourosh Khedri, Mohamed Abbas and Dr. Sameh Badrous.

Special thanks to Dr. Adel Shalaby - the author’s MSc advisor - who helped and guided him in building many of the research skills that helped him a lot during his PhD program.

The author also gratefully acknowledges the continuous Research Assistantship funding provided by Dr. Hoda ElMaraghy that gave him the opportunity to focus solely on his research.

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ACO | Ant Colony Optimization |
| AHP | Analytic Hierarchy Process |
| AMPL | A Mathematical Programming Language |
| ANP | Analytic Network Process |
| BOM | Bill of Materials |
| CAD | Computer-Aided Design |
| CAM | Computer-Aided Manufacturing |
| CNC | Computer Numerical Control |
| DSM | Design Structure Matrix |
| ERP | Enterprise Resource Planning |
| GA | Genetic Algorithm |
| GAPP | Generative Assembly Process Planner |
| GBOM | Generic Bill of Materials |
| GT | Group Technology |
| IDEF0 | A function modeling language, it stands for Icam DEFinition for Function Modeling, where ‘ICAM’ is an acronym for Integrated Computer Aided Manufacturing |
| IP | Integer Programming |
| KD | Knowledge Discovery |
| LCA | Least Common Ancestor |
| LCD | Liquid Crystal Display |
| LS | Local Search |
| M/C | Machine |
| MIP | Mixed-Integer Programming |
| MRP | Material Requirement Planning |
| MT | Master Tree |
| NP | Non-deterministic Polynomial-time |
| PDM | Product Data Management |
| PLM | Product Lifecycle Management |

| | |
|---|---|
| PSO | Particle Swarm Optimization |
| RF | Robinson-Foulds distance measure |
| RF-BOM | A modified version of Robinson-Foulds measure for comparing Bills of Materials |
| $\text{RF-BOM}_{max}$ | Maximum possible value for RF-BOM between two given BOM trees |
| $\text{RF-BOM}_{norm}$ | Normalized RF-BOM (between 0 and 1) |

# NOMENCLATURE

| | |
|---|---|
| $RC$ | Reconciliation cost |
| $N_D$ | Number of duplication events |
| $N_L$ | Number of loss events |
| $N_C$ | Number of extra items in the new BOM tree |
| $C_D$ | Unit costs for duplication events |
| $C_L$ | Unit cost for loss events |
| $C_C$ | Unit cost for having extra components in the new BOM tree |
| $N$ | Number of individual trees |
| $m1$ | Number of nodes (sub-assemblies) of tree 1 |
| $m2$ | Number of nodes (sub-assemblies) of tree 2 |
| $T1_{ij}$ | An element in the $i^{th}$ row and $j^{th}$ column of the encoded matrix representing tree 1 |
| $T2_{kj}$ | An element in the $k^{th}$ row and $j^{th}$ column of the encoded matrix representing tree 2 |
| $x1_{ik}$ | A binary (0-1) variable that takes 1 if node *i* in tree 1 is mapped to node *k* in tree 2. |
| $x2_{ki}$ | A binary (0-1) variable that takes 1 if node *k* in tree 2 is mapped to node *i* in tree 1. |
| $y1_{ik}$ | An auxiliary positive variable that represents distance from node *i* in tree 1 to node *k* in tree 2. |
| $y2_{ki}$ | An auxiliary positive variable that represents distance from node *k* in tree 2 to node *i* in tree 1. |
| $J_{ij}$ | Jaccard's component commonality measure (given by Equation 3.8) between pair of products *i* and *j*. |
| $n$ | Total number of different components |
| $d$ | Number of nodes of the master tree (given by n-1) |
| $m_i$ | Number of nodes in tree *i* |
| $t_{ijk}$ | A binary (0-1) element in the $j^{th}$ row and $k^{th}$ column of the encoded matrix representing the $i^{th}$ tree (e.g. the matrix in Figure 4.4) |
| $cm_{uj}$ | A binary (0-1) element in the $u^{th}$ row and $j^{th}$ column of the sought master matrix |
| $x_{uv}$ | A binary (0-1) element valued at 1 if node *v* is directly branched from node *u* in the master matrix |
| $y_{uj}$ | A binary (0-1) element valued at 1 if component *j* is directly branched from node *u* in the master matrix |
| $Z_{iku}$ | A binary (0-1) variable that takes 1 if node *k* in the individual tree *i* is to be mapped to the |

node $u$ from the master tree.

| | |
|---|---|
| $w_{iku}$ | An auxiliary positive variable that takes the value of the expression $\sum_{j=1}^{n}\left\|(t_{ijk} - cm_{uj})\right\|t_{i1j}$ (node-difference expression) when only $Z_{iku}$ is equal to 1. |
| $r_{ujv}$ | A linearization binary (0-1) variable that takes 1 if only the two binary variables $cm_{vj}$and $x_{uv}$ are both equal to 1. |
| $s$ | Total number of data instances (products or systems) |
| $u$ | Number of product features |
| $v$ | Number of manufacturing system capabilities. |
| $c_{ik}$ | A binary (0-1) element that takes value of "1" if capability $k$ exists in manufacturing system $i$, otherwise it is "0" (e.g. an element in the left hand side of Table 5.1) |
| $f_{ij}$ | A binary (0-1) element that takes value of "1" if product feature $j$ exists in product $i$, otherwise it is "0" (e.g. an element of the right hand side of Table 5.1) |
| $b_{km}$ | A binary (0-1) element that takes value "1" if the two capabilities $k$ and $m$ should not be associated together with the same feature. |
| $a_{kj}$ | A binary element in the matrix in Table 5.2 (association matrix); $a_{kj}$ takes the value "1" if the manufacturing capability $k$ is associated with product feature $j$ and "0" otherwise. |
| $h_{ij}$ | An auxiliary binary variable to relax constraint 5.2' (at indices $i$, and $j$) when inconsistent data is present. |
| $g_{ik}$ | An auxiliary binary variable to relax constraint 5.3' (at indices $i$, and $k$) when inconsistent data is present. |

# CHAPTER 1
INTRODUCTION

## 1.1 Motivation

In today's modern manufacturing environment, globalization, unpredictable markets, increased products customization and the pursuit for competitive advantages are some of the many challenges facing manufacturing enterprises (ElMaraghy 2009). Such challenges are triggering more frequent changes in product design as well as increasing need for variety, leading to significant impact on corresponding design, planning and manufacturing costs. The ability of a manufacturing enterprise to efficiently adapt to these conditions has become a pre-requisite for its survival.

Product development involves all the activities and processes required to get a new product to the market. Product design, manufacturing and assembly are among the core activities of product development. A Large volume of data is accumulated over time in industrial databases (e.g. Enterprise Resource Planning (ERP) and Product Life Cycle Management (PLM) systems) at various stages and levels of product development. There is valuable hidden knowledge and patterns embedded in this data. Discovering and interpreting such knowledge could be very useful in supporting and expediting companies planning and operation activities. Extracting useful knowledge from a given set of data is a commonly encountered task in many research fields. Several expressions are used to refer to this task, such as knowledge discovery, knowledge extraction, data pattern processing, information harvesting, information discovery, and data mining. Knowledge Discovery (KD), the preferred expression in this research, is a very wide topic of interest, primarily to Information and Computer Science researchers; however, is does also have several design and manufacturing related applications (Kusiak 2006; Choudhary et al. 2009; Hui 2011; Ruo and Fu 2011). The term knowledge discovery was first introduced by Piatetsky-Shapiro (1991). Knowledge discovery could be defined as the non-trivial process of recognizing valid, novel, potentially useful, and eventually understandable patterns in data (Fayyad et al. 1996).

KD is a wider concept than *Data Mining*; the more popular term especially among Industrial Engineering, Statistics, and Information Systems research communities. According to (Fayyad et al. 1996), Data Mining could be considered as the main step in the KD process which is concerned with applying data analysis and discovery algorithms to generate a particular enumeration of patterns (or models) over the data; while the KD may involve, in addition to Data Mining, more steps such as data selection, preprocessing, transformations, and interpretation/evaluation of results. Clustering, classification, regression, summarization, association, and detection are the major subjects for knowledge discovery (Fayyad et al. 1996).

In light of that, this research is in fact motivated by the growing need for more practical and efficient knowledge-based methods that could help manufacturers be more responsive and cost-effective by making the best use of the design and manufacturing data stored in databases for existing or old products. Significant research efforts have been carried out in that direction; however, as will be shown in the following chapters throughout the dissertation, a large space for new ideas and methodologies as well as enhancements of existing ones is still available and needs to be better exploited.

## 1.2 Research Scope

This research is addressing the following three main applications for how knowledge discovery in design and manufacturing data could be of a great benefit for planning for new products and different phases of the development process:

- *Product Design Retrieval*
- *Assembly Sequence Planning*
- *Manufacturing System Synthesis*

In the next sub-sections, the description of the problem targeted in each of these applications is outlined. Besides, a further problem (*Product Family Formation*) is also described, as being an extra application for the approach used for product design retrieval. Product design retrieval, assembly sequence planning and product family formation problems addressed in this research are concerned with mechanical/electronic assembly products (e.g. an electric motors or a coffee maker); while manufacturing system synthesis is for both assembly products as well as products that are just made of single parts (e.g. connecting rod of an automobile engine or the housing of a gear pump). The models and methods proposed in this research could be considered as a package of knowledge-based solutions that can greatly support product designers and manufacturers at various stages of the product development process.

Before describing the addressed application, it is worth mentioning first what it is meant by the terms data and knowledge in the context of this dissertation. In fact, *data* is considered as the input raw material by which a knowledge discovery model utilizes and processes so as to eventually produce a meaningful ready-to-use product which is *knowledge*. As mentioned by W. ElMaraghy (2009), knowledge is a high-value form of information that is ready to apply to decisions and actions. Having a lot of data about any given design or manufacturing process without utilizing it using the proper knowledge extraction method is like having several scattered LEGO blocks that would not have a meaning or a purpose unless they are properly put together.

*1) Product Design Retrieval*

Retrieval of legacy product designs and their associated design and manufacturing data is one of the effective tools used for rapid product development by capitalizing on existing similarity between new and legacy designs. In this research new product variants as well as old ones are represented by their Bill of Materials (BOM) trees in which it is required to find the old product with the BOM tree that best matches the BOM tree of a new product.

Upon finding that best matching old product, associated design data such as the corresponding CAD models, tolerances, materials and other design considerations of the old product, can be used and modified to fit the new product. Hence, the time required to design the new product would be reduced. In addition, other useful data such as those related to the assembly plan and supply chain can also be retrieved and utilized for the new product.

*2) Product Family Formation*

It is important for a manufacturing system to identify sets of products that share significant manufacturing resources and setup and process them together. This shall maximize manufacturing system utilization and productivity and decrease its complexity. Accordingly, the problem of grouping several different product variants (e.g. models or types) of a given product into number of groups or families based on similarity in components and configuration (i.e. BOM) was addressed.

*3) Assembly Sequence Planning*

Final assembly of products such as electronic products (PCs and tablets) or mechanical products (pumps and motors) need to be planned for the sequence by which their parts are to be assembled together. This is known as assembly sequence planning. The assembly sequence has a great effect on the cost and efficiency of the entire assembly process. Finding an optimal, or even a fairly good assembly sequence, is not a simple task, in which the number of possible sequences could be very large even for a small number of parts.

A shortcut that many manufacturers use is to retrieve the assembly sequence of the most similar old product variant they have in their records, and then use it, with necessary modifications, for the new product variant. However, this approach does not work well if the new variant has a combination of parts that never existed together in any of the old variants, even if the records include data about all parts of the new product variant. This is what the novel knowledge-based assembly sequence planning approach proposed in this research is addressing.

*4) Manufacturing System Synthesis*

Manufacturing systems synthesis activities involve specifying the production capabilities including manufacturing processes; machines and equipment that are usually selected from pre-defined sets to produce given product variants (AlGeddawy and ElMaraghy 2011). It would be very useful to use manufacturing records of existing and old products to establish a mapping between elements of the product domain on one side, such as product features and/or product modules/components, and elements of the manufacturing system domain on the other side, such as manufacturing capabilities or machines/equipment.

By having enough data instances about previous product designs and corresponding manufacturing systems used to produce them; this research establishes such mapping and uses it for manufacturing system synthesis on the machine capabilities level. The mapping results would be in the form of what particular manufacturing system capabilities are required for producing a given product feature.

## 1.3 Research Literature Gaps

The research work previously done in each of the above mentioned product development applications studied in this research will be thoroughly reviewed in its corresponding chapter. However, it is useful to highlight in this introductory chapter the research gaps in each application.

Few methods have been developed in literature, using graph difference operations, linear algebra and Integer Programming (Romanowski and Nagi 2005; Romanowski et al. 2006; Shih 2011; Gongzhuang et al. 2014), for matching trees of BOM and finding pairwise similarity measures for retrieval applications of design and manufacturing knowledge. These methods take into consideration the matching of the intermediate BOM tree nodes which are believed to be of no significance and might even produce misleading results. A BOM intermediate node (sub-assembly) is indeed equivalent to its sub-nodes (components). The only factor that should matter for trees matching is the difference in topologies and contents of end nodes. Furthermore, existing methods are computationally intensive and their ability to handle real problems of large sizes is limited.  Well-developed tree reconciliation algorithms, used in Biological Science fields, are utilized in Chapter 2 to develop an alternative method that is more robust and computationally efficient for BOM tree matching.

Many product family formation methods have been developed for forming product families (Galan et al. 2007; Eguia et al. 2011; Pattanaik and Kumar 2011; Goyal et al. 2013; Gupta et al. 2013) based on commonality of components, as well as other similarity measures such as reusability, operations sequence and demand. None of the existing grouping methods takes into consideration similarity in products structure. It is not uncommon to find products which share many common components but with considerably different assembly structure. For example, a group of water kettles may share the same power switch, but at different position in each kettle variant (e.g. at the top or at the bottom). From an assembly point of view, more convincing products grouping (family formation) results should be obtained if product structure is taken into consideration. The proposed product family formation method in Chapter 3 based on BOM trees matching is addressing this limitation.

Existing methods for retrieval or knowledge-based assembly sequence generation rely on retrieving the most similar existing product variant with respect to commonality of product components and assembly structure (Dong et al. 2005). The assembly sequence plan of such a retrieved product variant is then used as a preliminary one for the new variant. The assembly sequence data (i.e. sequencing relationships between components) for a new combination of components might be completely available in the manufacturer's database but in a scattered form, where overlapping sub-groups of that new combination of components are distributed among different product variants. Thus, the critical limitation of individual assembly sequence retrieval methods is that the assembly sequence data for a given new combination of components that has never existed together before in an existing variant could not be captured. This is what the proposed knowledge-based assembly sequencing approach in Chapter 4, based on the novel concept of master/consensus assembly sequence tree, has been designed to overcome.

The proposed manufacturing systems synthesis in Chapter 5 is based on mapping between design and manufacturing domains using association rule discovery based on existing or historical data. Association rule discovery is an important Knowledge Discovery topic that was popularized particularly due to the prominent work of Agrawal et al. (1993) who studied mining a large collection of basket data transactions for significant association rules between sets of items (products). Association rule discovery algorithms based on the work of Agrawal et al., which is the typical tool used by most researchers practitioners in the field, generates very large number of redundant rules where a lot of post processing work is needed to interpret meaningful rules out of the large number of generated rules. Furthermore, properly defining minimum statistical thresholds to consider a given rule as a significant rule is a critical decision to which the obtained results are quite sensitive. Similar to the cladistic-based model of AlGeddawy and ElMaraghy (2011), the association rule discovery model proposed in Chapter 5 is using a non-statistical approach based on Integer Programming (IP). From an implementation point of view, the proposed IP model is less complex.

## 1.4 Thesis Statement

The thesis statement of this research could be formulated as follows:

*Innovative and well-designed knowledge discovery models could simplify and enhance the process of extracting useful patterns and knowledge from existing and old product design and manufacturing data.*

The simplification term in this statement mainly refers to implementation aspects (i.e. ease of implementation as well as computational efficiency), while the enhancement term refers to the ability to extract knowledge that could not be extracted by existing methods.

## 1.5 Overview of Developed Methods and Case Studies

A novel approach is introduced in this research for each of the above mentioned problems, new methods were developed and new applications in engineering of existing successful tools found in other fields are also presented. In the following sub-sections an overview of these methods is offered.

*1) Product Design Retrieval*

A new automatic design retrieval method which identifies the legacy product design most similar to a new one is proposed. Matching phylogenetic trees has been utilized in biological science for decades and is referred to as “tree reconciliation”. A new application of this approach in manufacturing is presented where legacy designs are retrieved based on reconciliation of trees representing products Bills of Materials. A product Bill of Materials (BOM) is a structured tree which represents its components and their hierarchal relationships. A chemical processing centrifugal pump is used as a case study for illustration. The results obtained using the proposed method is compared with those recently published on BOM trees matching for further analysis and verification. This novel method is less computationally complex than available state-of-the-art algorithms.

*2) Product Family Formation*

In this research, a novel Bill of Materials (BOM) trees matching Integer Programming model is introduced to addresses commonality of components as well as their hierarchical assembly structure. A product Bill of Materials (BOM) trees matching model, inspired by well-established techniques used in biological fields for comparing phylogenetic trees, has been developed. Hierarchical clustering is applied to form groups of product families based on the pair-wise similarities obtained by matching BOM trees. The proposed method is applied to a case study of six centrifugal pumps for demonstration and analysis. More assembly-oriented products grouping outcome could be obtained using the proposed method compared with that obtained based on typical component commonality measures.

*3) Assembly Sequence Planning*

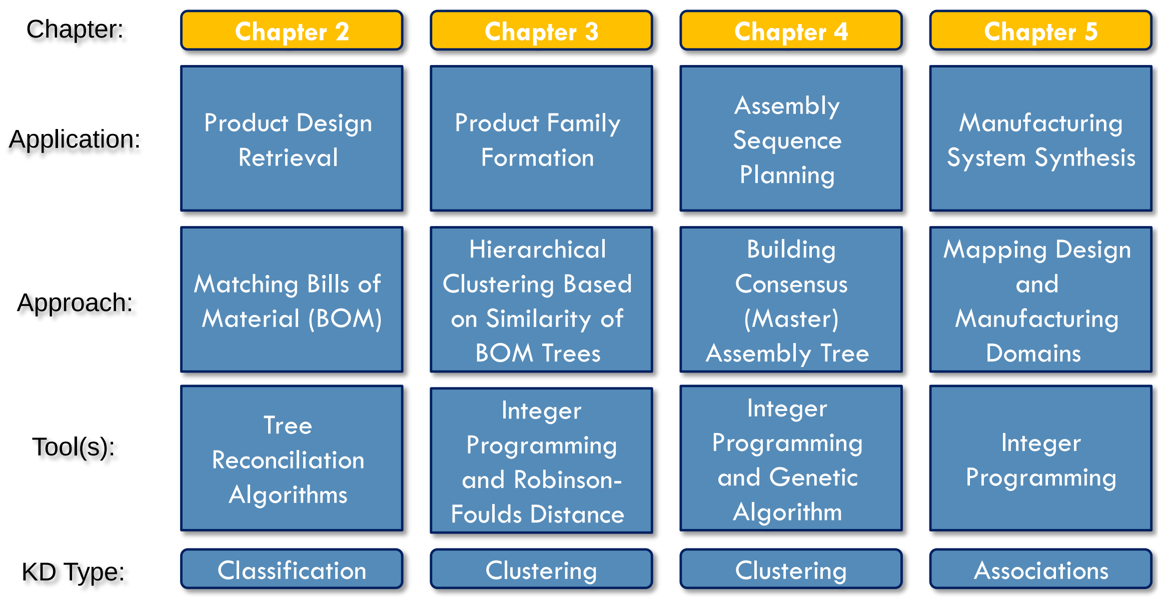
A novel knowledge-based Mixed-Integer Programming (MIP) model is presented for generating the assembly sequence of a given product based on available assembly sequence data of similar products. The proposed mathematical model finds the optimal consensus assembly sequence tree for an existing product family based on the assembly sequence trees of individual product family members. The model is inspired by problems studied in phylogenetics and evolutionary studies where the consensus species classification for a set of conflicting classifications is sought. The generated consensus tree serves as a master assembly sequence tree and is used to generate the assembly sequence for a new product variant that falls within, or significantly overlaps with, the boundary of the considered family of products. The developed model is demonstrated using a family of pilot valves. Furthermore, in addition to the developed MIP model, a Genetic Algorithm (GA) was also developed to construct the master assembly sequence tree. The GA-based method could be used for problem sizes that could not be handled by the MIP model in a reasonable time.

*4) Manufacturing System Synthesis*

A novel knowledge discovery model is introduced to extract useful correlations between the manufacturing domain and the design domain based on historical manufacturing and design data. Given the data on sets of products and the corresponding manufacturing systems used in their production, an Integer Programming model is developed to extract association rules between various product features and manufacturing capabilities. These associations identify the specific manufacturing system capabilities associated with the production of each product feature. Such a kind of discovered knowledge could be used to synthesis the required manufacturing system capabilities for prospective products of new combinations of features. The proposed model is applied to two case studies; the first is for machined parts adopted from the literature. The second case study if for engine block assembly that demonstrates the applicability of the model to assembly systems synthesis in addition to its applicability to machining system synthesis. From an implementation point of view, the proposed IP model is more straightforward than existing methods.

## 1.6 Thesis Structure

As mentioned in the research scope in Section 2.1, the models and methods presented in this research is a package of knowledge-based solutions that can greatly help product designers and manufacturers save a lot of effort and time at various stages of the product development process. Figure 1.1 shows the structure and flow of the research carried out in this dissertation; at the upstream stage of the product development process Chapters 2 and 3 are addressing the design retrieval and product family formation problems. Chapter 4 is then addressing the assembly sequence planning problem which is the next major stage in the product development process. Chapter 5 is finally addressing the manufacturing/assembly system synthesis problem (i.e. equipment selection) which comes at the downstream stage.



| Chapter: | Chapter 2 | Chapter 3 | Chapter 4 | Chapter 5 |
|---|---|---|---|---|
| Application: | Product Design Retrieval | Product Family Formation | Assembly Sequence Planning | Manufacturing System Synthesis |
| Approach: | Matching Bills of Material (BOM) | Hierarchical Clustering Based on Similarity of BOM Trees | Building Consensus (Master) Assembly Tree | Mapping Design and Manufacturing Domains |
| Tool(s): | Tree Reconciliation Algorithms | Integer Programming and Robinson-Foulds Distance | Integer Programming and Genetic Algorithm | Integer Programming |
| KD Type: | Classification | Clustering | Clustering | Associations |

**Figure 1.1**: Research map

The dissertation is presented in 6 Chapters where the literature review related to each topic is presented at the beginning of the corresponding chapter. In Chapter 2 the proposed BOM trees matching method based on tree reconciliation algorithms used in Biological Sciences is presented. Chapter 3 presents product family formation as a further application for matching BOM trees in addition to product design retrieval. Chapter 4 presents the knowledge-based assembly sequencing method based on the notion of consensus trees. Chapter 5 presents the Integer Programming association rule discovery model developed for constructing a mapping between product design domain and manufacturing system domain to be then used for manufacturing/assembly system synthesis. Finally, the summary and conclusions as well as the future work of this research are presented in Chapter 6.

# CHAPTER 2
PRODUCT DESIGN RETRIEVAL BY MATCHING BILL-OF-MATERIAL (BOM) TREES

## 2.1 Introduction

The first problem addressed by the proposed package of knowledge discovery models is product design retrieval. Automatic retrieval of relevant legacy product designs and associated design and manufacturing data is one of the major tools used for rapid product development by capitalizing on existing similarity between new and legacy designs. Matching BOM trees and finding pairwise similarity measures between them is a "classification" type for knowledge discovery that has been recently used only for retrieving design and manufacturing data. It could be seen as a new alternative to conventional Group Technology (GT) classification and coding systems (Opitz 1970) which are used for clustering and design retrieval (Barton and Love 2005). An important difference is that classification and coding systems and most other design retrieval methods, such as CAD-based (Kunpeng et al. 2012) and image processing-based methods (Lim et al. 2006), are mainly concerned with parts not products, based on similarity of their geometric features and technological specifications. Bills of Materials matching addresses the assembly level, where similarity between whole products is sought based on the product structure and its components.

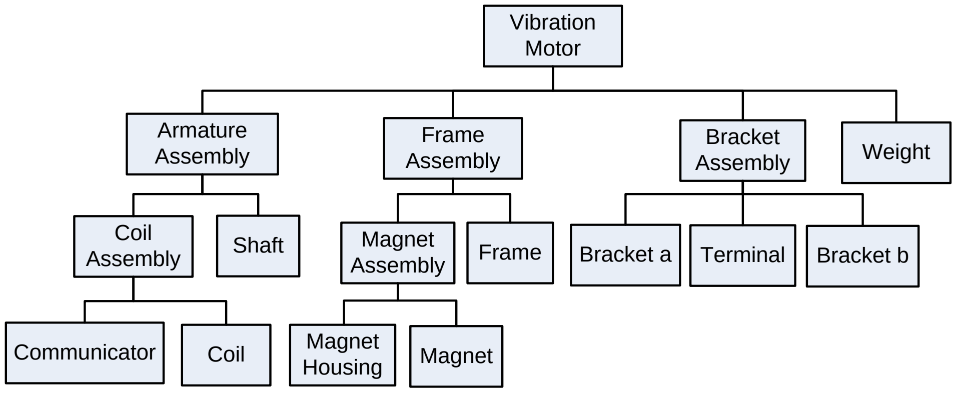


**Figure 2.1**: BOM tree for a vibration motor used in cell phones (reproduced from (Jiao et al. 2008))

The Bill of Materials (BOM) was first introduced by Orlicky (Orlicky 1971; Orlicky et al. 1972) as a product data structuring form for Material Requirements Planning (MRP) systems, commonly used in production planning and inventory control systems. A BOM is the list of sub-assemblies, components, parts, raw materials and the quantities required of each to produce an end product. Unordered rooted trees are used to represent BOMs (Orlicky 1971). Figure 2.1 shows a simple three-level BOM tree for a vibration motor used in cell phones; however, a BOM of a complex product would have more components and hierarchical levels.

Upon finding the most similar BOM of an existing product to the BOM of a new product, associated data, such as CAD models, tolerances and assembly process plan of the identified existing product, can be used and modified to fit the new one. Hence, the time required to develop the new product would be reduced. In addition, other useful data such as those related to the product supply chain can also be modified and utilized for the new product. In reality, BOM data could be extracted from the Material Requirement Planning (MRP) module of the Enterprise Resource Planning (ERP) systems, such as ECi M1, Infor VISUAL ERP and E2 shop, used by production facilities to collect, store and manage data from many business activities including product planning and control, supply chain and inventory management.

This chapter proposes a new application of tree reconciliation techniques used in biological sciences, to be described later, and tailor it for matching products Bills of Materials (BOM). Tree reconciliation algorithms were shown to be computationally efficient and suitable for matching BOM trees.

## 2.2 Literature Review

In addition to their classical use in Material Requirements Planning (MRP), BOM trees have other valuable applications. For instance, Jiao et al. (2000) proposed an integrated product and production data management method for mass customization systems. They proposed a production data structure called Bill of Materials and Operations (BOMO) for unifying BOMs and routings in order to facilitate better production planning and control, order processing, and engineering change control. Steva et al. (2006) employed BOMs along with other tools such as Design Structure Matrix (DSM) and Function Diagrams to develop a product platform identification methodology. Other applications for BOM in supporting product modeling and variety management, such as in supporting assemble-to-order manufacturing (Van Veen and Wortmann 1987) and end-of-life decision making in Design for the Environment (Gonzalez and Adenso-Diaz 2005) have been reported. BOM trees matching is differentiated as a design retrieval method from conventional methods such as Group Technology (GT) (Barton and Love 2005), (Lee-Post 2000), CAD (Bai et al. 2010), (Kunpeng et al. 2012), Neural Networks (Chieh-Yuan and Chang 2003), and image processing based methods (Lim et al. 2006) because it addresses the product assembly level, where similarity between products as a whole, not just individual parts, is sought based on similarity of their components and assembly structures.

Bill of Materials (BOM) trees are classified as unordered rooted trees. At any given level of a BOM tree hierarchy, the order of nodes belonging to the same node has no significance. For instance, it makes no difference to say that a given pen has a tube, tip and cap, or it has a tip, cap and tube. The first level nodes of a BOM tree are all belong to a single node which is the root. Accordingly, classical tree-matching algorithms commonly used in Mathematics and Computer Sciences (Shasha et al. 1994; Jiang et al. 1995; Chawathe and Garcia-Molina 1997) could be used to compare BOM trees. Most of these algorithms are based on performing limited sets of editing operations to transform two trees into isomorphic (i.e. congruent) form. A unit cost is associated with each editing operation and the objective is to find the set of editing operations which represents minimum total matching cost. These editing operations include node deletion, node insertion and node substitution.

Romanowski and Nagi (2005) presented a critical criticism of the editing-based trees matching approaches and showed, using examples, that these algorithms may yield incorrect results when used to match trees of BOM. They introduced the first tree-matching algorithm specific to BOM trees and named it Decomposition-Reduction or DeRe algorithm, which is based on the symmetric difference operation used in set theory. In a tree matching context, symmetric difference is the set of tree edges that appears in one tree but not the other. Edges in BOM represent parent-child relationships; thus, the symmetric difference between two BOM trees represents the dissimilarity in both content and topology. DeRe algorithm uses a modified weighted symmetric difference operation in which partial matching between edges is allowed. Accordingly, the algorithm must be provided with the distances/differences between every pair of components (tree leaves), while distances between sub-assemblies (tree nodes) are derived from differences between components. Thus, the main task is to find the minimum weighted symmetric difference. Romanowski and Nagi (2004) used the DeRe algorithm along with a trees combining algorithm to build a Generic Bill of Materials (GBOM) and applied it to more than 400 variants of nurse call devices.

Romanowski et al. (2006) reported some limitation of the DeRe algorithm, formerly introduced by two of the authors (Romanowski and Nagi 2004). They mentioned that DeRe, while being more accurate than conventional editing-based techniques, may in some cases overstate the dissimilarity measure. Accordingly, they formulated an Integer Programming model that basically maximizes, for a given pair of BOM trees, the matching score between every possible sub-tree of a given tree, and those of the tree being matched. The model was proven to be NP-hard, which means that heuristic algorithms would be required to solve real-sized datasets. Some solution approaches were suggested without reported implementation.

In Shih (2011), BOM trees matching has been treated as an orthogonal Procrustes problem, which is a matrix approximation notion in linear algebra. Given two matrices $X$ and $Y$, it is required to find an orthogonal transformation matrix $T$ that most closely maps $X$ to $Y$. The simplest algebraic statement of a Procrustes problem is finding matrix $T$ that minimizes $\|X - YT\|$. Any two BOM trees are represented by adjacency matrices, then a transformation matrix $T$ is found and used to calculate a similarity measure between the two adjacency matrices. Different importance weights were assigned to each level of a given BOM, where higher tree levels are assigned higher weights. The algorithm involves several matrix multiplication operations which makes it computationally intense, especially for large size matrices.

In conclusion, a few techniques exist for matching trees of BOM trees; however, they take into consideration the intermediate nodes which are believed to be of no significance and might even give misleading results. A BOM intermediate node (sub-assembly) is indeed equivalent to its sub-nodes (lower sub-assemblies and/or components). The only factor that should matter for trees matching is the difference in topologies and contents of end nodes. Well-developed tree reconciliation approaches, used in biological science fields, do satisfy this objective and are computationally more efficient.

## 2.3 Scope

Bill of Material (BOM) is originally defined and formatted as shown in the introductory section of this chapter, however it could have other configurations, levels of detail, amount and type of information that depend on the production stage as well as application area (Stonebraker 1996). Variants of Bills of Material may include: Engineering BOM (EBOM) (Kneppelt 1984), Planning Bill of Material (PBOM) (Maskell 1988), Operations Bill of Material (OBOM) (Mather 1987), Manufacturing Bill of Material (MBOM) (Chang et al. 1997), or other customized forms optimized for  specific applications exist. Hence, there is no unified format for Bill of Material and several BOM formats may be used for the same product to suit its application in its different production stages.

The method proposed in this chapter to match Bills of Material uses Bill of Material trees as defined in Section 2.1, in which it captures subassemblies and components of the product, as well as their quantities and hierarchical assembly relationships, used by design engineers to represent designed product structure. It should be also mentioned that for a given new product design with new functional requirements, the design stage by which the proposed BOM-based retrieval method is to be involved is the embodiment design stage, which is an intermediate stage that comes after the conceptual design stage and before the detailed design stage. At the conceptual design stage, a BOM tree for the new product that satisfies its new functional requirements may not be available yet. The detailed design stage should then come after the BOM-retrieval (embodiment stage) in which modifications and tuning of the retrieved product design to match the new design requirements should take place. The term “subassembly” is used in this chapter instead of the closely related term “module”, where a module is in fact a special type of subassemblies that should typically have one or more well-defined functions by itself (Miguel 2005) (e.g. sheets feeding module of a photocopier), which is not necessarily the case for what is meant here by a subassembly. The term “product structure” is also used in this research instead of the term “product architecture” which is defined as the way in which the functional elements of a product are arranged into physical units and the way in which these units interact (Ulrich and Eppinger 2011). Product architecture is a broader and more functional-oriented term than the term product structure used in this research to refer to assembly-related aspects of products.

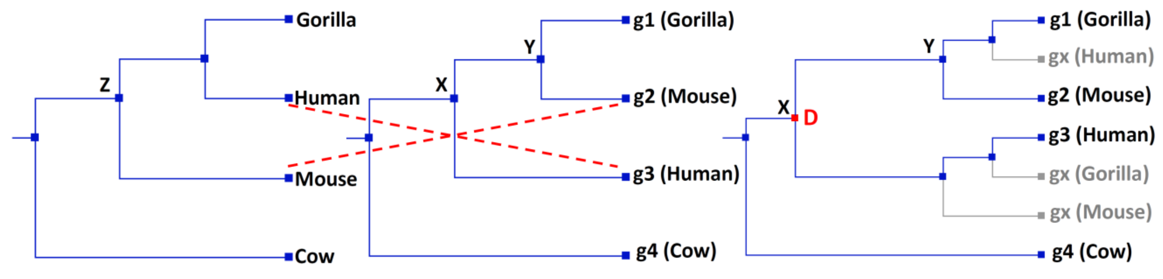
Subassemblies and components in BOMs are usually identified by numbers or codes in order to differentiate between several versions or variants of the same subassembly or component. Dealing with each version or variant of a given component as a distinctive component will make it very uncommon to have matching components, and hence, will not allow for a successful BOM matching. Thus, it is assumed that considerably similar versions and variants of each component or sub-assembly are grouped first under the same title or identification code.

It is assumed in this research that a consistent methodology is followed in constructing all existing BOMs. It is also assumed that the same language/naming and consistent terminology are used in labeling elements of existing BOMs as well as the new BOM. The computational efficiency of the presented method, as will be discussed later in Section 2.4.2, as well as the availability of appropriate software packages, makes it scalable and capable of handling large size data sets with numerous components and several hierarchical levels.

## 2.4 Tree Reconciliation

### 2.4.1 Background

The evolutionary history and classifications of organisms in Biological Science fields is commonly represented by trees known as phylogenetic trees that typically are unordered rooted trees. Tree reconciliation is concerned with the proper mapping between two phylogenetic trees (known as cladograms): an associate tree (e.g. gene family tree) and a host tree (e.g. species tree) with the aim of properly reconstructing the evolutionary history for the associate tree (Page 1994). Such a reconstruction process is usually based on the assumption that some biological events; such as gene *duplication* and *loss* events (to be explained later), are not properly recorded on the existing form of the associate tree and, hence, they need to be algorithmically postulated.



**(a) Species Tree** **(b) Original Gene Tree** **(c) Reconstructed Gene Tree**

**Figure 2.2**: An example for reconstructing a gene tree using tree reconciliation (based on one of NOTUNG software examples (Danicic et al. 2008))

Figure 2.2 shows a simple example of what is meant by tree reconciliation. The first tree (Figure 2.2(a)) is a cladogram that classifies the four species *Gorilla*, *Human*, *Mouse* and *Cow*; while the second tree (Figure 2.2(b)) is the cladogram classifying the four genes *g1*, *g2*, *g3* and *g4* collected from the four species in the first tree. The labels on the gene tree include, in between brackets, the names of the species from which each gene is obtained. It is clear that both trees do not match since the arrangement of species (end nodes) on both trees is different; hence, a new cladogram for the genes is reconstructed to rectify that. The third cladogram (Figure 2.2(c)) is the gene tree reconstructed using tree reconciliation; the reconstructed gene tree contains the original gene tree as a homomorphic (structure-preserving) sub-tree and in the meantime the arrangement of species on the reconstructed gene tree does not contradict with the arrangement of species on the species tree (Bonizzoni et al. 2005).

In this example, one gene duplication event is postulated at gene *X* where a capital letter D is shown at that point in the reconstructed gene tree, as well as three gene loss events (pale branches). Gene duplication event means that two copies of a given gene start changing independent of each other and both are being inherited by subsequent generations. A gene loss event (represented in the figure as *gx*) refers to genes that are either missing or not yet discovered; however, the species from which each missing gene should originate are known according to this analysis. Typical applications for tree reconciliation are gene function prediction, gene annotation, and identifying drug targets (Searls 2003; Bourgon et al. 2004; Gabaldon 2006).

### 2.4.2 Tree Reconciliation Algorithm

Many tree reconciliation algorithms and software packages are now available such as GeneTree (Page and Charleston 1997), Softparsmap (Berglund-Sonnhammer et al. 2006) and NOTUNG (Chen et al. 2000). The last version of the later, NOTUNG 2.6, is utilized in the research reported in this chapter. Based on the work in (Chen et al. 2000; Durand et al. 2006; Vernot et al. 2008), duplication-loss reconciliation algorithms work as follows. Given a gene tree $T_G$ and species tree $T_S$, it is required to identify all gene *duplication* and *loss* events that bring $T_G$ into congruency with $T_S$. Accordingly, it is required to construct a mapping *M(v)* for each gene *v* in $T_G$ to a corresponding species in $T_S$. For genes at the leaves of the tree, each gene is mapped to the species on the species tree from which the gene was obtained. As for other genes (ancestral genes), a given gene *v*, with left and right direct descendant genes *l(v)* and *r(v)*, is mapped to the least common ancestor (*LCA*) of the two species mapping *l(v)* and *r(v)* in the species tree as shown in Equation 2.1 (Chen et al. 2000).

$$M(v) = LCA(M(l(v), M(r(v))) \quad (2.1)$$

Hence, a given gene *v* in the original gene tree is to be considered as a duplication node in the reconstructed gene tree if, and only if, it is mapped to the same species, by which any of its direct descendant genes is mapped: *M(v) = M(l(v)) or M(v) = M(r(v))*. For instance, in Figure 2.2, the mapping of both the gene *X* and its direct descendant gene *Y* in the original gene tree is the same species *Z* in the species tree, thus gene *X* is a duplication node as shown on the reconstructed gene tree.

As for gene loss events, Equation 2.2 (Chen et al. 2000) is first used to calculate the number of losses associated with each edge *e(u, v)* connecting gene *u* and its direct ancestor *v* in $T_G$. Where *Depth(M(u))* and *Depth(M(v))* in the equation returns the hierarchal level of the species mapping genes *u* and *v* in $T_S$ and *IsDup(v)* is a binary variable that returns 1 if *v* is a duplication node and zero otherwise. The species associated with the losses identified at *e(u, v)* are determined by searching the species tree along the path from *M(u)* to *M(v)*. For each species *s* between *M(u)* and *M(v)*, a lost gene is assigned to the left or right direct descendant species of *s*; whichever is not appearing on the path from *u* to *v* in the gene tree. By applying Equation 2.2 for instance on the edge *e(g1, Y)* connecting gene *g1* and *Y* in the gene tree in Figure 2.2(b) we get a number of loss events equal to 1 (4 – 2 - 1 + 0). The species from which the gene is lost should be *Human* as it is the missing species on the edge *e(g1, Y)*.

$$Losses(e(u, v)) = Depth(M(u)) - Depth\ (M(v)) - 1 + IsDup(v) \quad (2.2)$$

According to this algorithm the reconstructed gene tree should have the least postulated number of *duplication* and *loss* events (most parsimonious cladogram). The total cost of reconciliation (matching cost) of two trees $T_G$ and $T_S$ is then given by Equation 2.3 (Chen et al. 2000), where $N_D$ is the number of duplication events and $N_L$ is the total number of loss events, while $C_D$ and $C_L$ are unit costs for duplication and loss events respectively.

$$RC(T_G, T_S) = C_D N_D + C_L N_L \qquad (2.3)$$

The above algorithm applies when both the gene and species trees are binary meaning that each gene or species has exactly two direct descendants. It is a very computationally-efficient algorithm that is based on linear time routines. The algorithm was extended in (Vernot et al. 2008) to handle two trees one of them is non-binary. Although the extended algorithm involves more computations, it does still have very reasonable computational complexity. The kernel of the extended algorithm is the duplication identification routine which is of a quadratic complexity and it was reported in (Vernot et al. 2008) to be able to reconcile 1174 test trees in 48 seconds. Complexity of those trees (i.e. number of components and hierarchical levels) is not reported in that study; however, even if those trees were of low complexity, reconciliation of that many trees in such a short time is a strong indication to the computational-efficiency of tree reconciliation algorithms.

## 2.5 BOM Tree Reconciliation

Fundamentals of a typical tree reconciliation algorithm have been presented in Section 2.4. It can fit, with some adjustments, the requirements of matching BOM trees. Accordingly, lower reconciliation cost between two given trees implies less structural differences, which is equivalent to better matching. In view of that, a given new BOM tree could be treated as a species tree and each existing BOM tree in the database could be treated as a gene tree. Each existing BOM tree is reconciled with the new BOM tree and a reconciliation cost is calculated using NOTUNG 2.6. Hence, the existing tree with least reconciliation cost should be the best matching tree to the new one as illustrated in the following subsections. It is important to emphasize that the explained mathematical manipulation of the trees is carried out to identify the closest matching trees and penalize differences. However, such mathematical manipulation does not change the trees being compared, their components and sub-assemblies or the hierarchical relationships between them.

### 2.5.1 Handling Non-Binary Trees

Current tree reconciliation algorithms are capable of considering binary trees only for both gene and species trees, where each node has at most two sub-nodes. However, some algorithms, including the one implemented in NOTUNG (Danicic et al. 2008), are capable of reconciling binary gene tree to a non-binary species tree, or a non-binary gene tree to a binary species tree. It is very common for BOM trees to exist as non-binary trees; however, converting a non-binary tree into a binary tree is a straightforward operation (Figure 2.3). Thus, before performing reconciliation of any two non-binary BOM trees, one of them should be first converted into a binary tree.

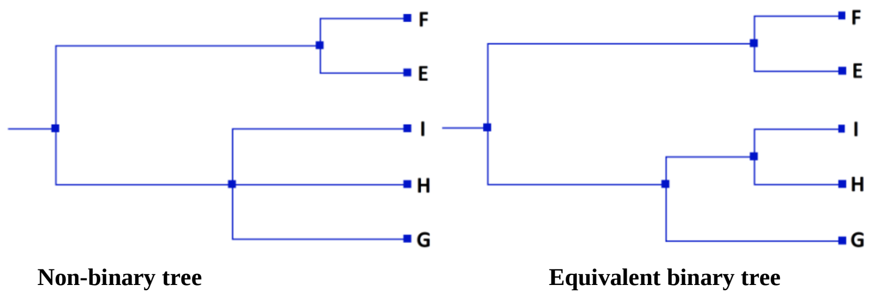


**Figure 2.3**: Converting a non-binary tree into an equivalent binary tree

The retrieval of the most similar legacy BOM tree to a new one requires pairwise tree reconciliation between a new BOM tree on one side and several existing BOM trees on the other side. Accordingly, before reconciling any existing BOM (binary or non-binary) to a non-binary new BOM tree, the new BOM tree has to be converted first into a binary tree; it is more practical to convert only the new BOM tree instead of converting several existing ones. Having at least one binary tree out of any two trees to be reconciled is just a prerequisite for the reconciliation algorithm to work.

### 2.5.2 Extra Components in the Existing BOM Trees

For any tree reconciliation algorithm, all items of the gene tree must also exist in the species tree; however, the species tree may include extra items (end nodes). From a BOM trees matching perspective, those extra components are indeed a source of dissimilarity that should be accounted for. Therefore, as a preparation step, before applying the reconciliation algorithm, the existing BOM tree to be considered for reconciliation, with a new BOM tree, is first searched for any items that do not exist in the new BOM tree. During reconciliation, this set of items is added as pseudo items to the new BOM tree; thus, for each existing BOM tree a modified version of the original new BOM is created. The example in Figure 2.4 shows three trees; the first from the left is a new BOM tree, the second is an existing BOM tree, while the third is the modified new BOM tree after adding a new branch of pseudo items (out group). The new branch includes the two items *K* and *L* found in the existing BOM tree but not in the new one.

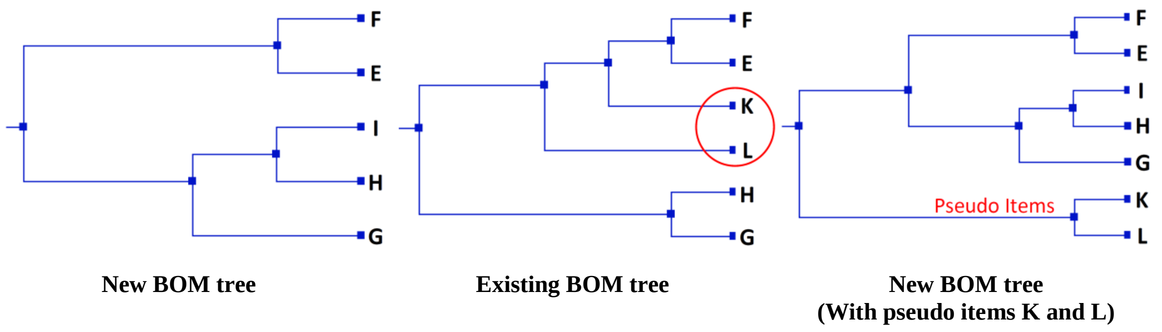


**Figure 2.4:** Adding extra components in existing BOM trees (pseudo items) to the new BOM tree

### 2.5.3 Extra Components in the New BOM Tree

Extra items in the new BOM tree which do not exist in the existing trees are a source of dissimilarity that needs to also be considered. In order to account for this type of dissimilarity, the number of extra items in the new BOM tree becomes an additional term in the total reconciliation cost (Equation 2.3). Accordingly, the total reconciliation cost is now given by Equation 2.4, where $N_C$ is the number of extra items in the new BOM tree, and $C_C$ is its corresponding unit cost.

$$RC(T_G, T_S) = C_D N_D + C_L N_L + C_C N_C \qquad (2.4)$$

### 2.5.4 Difference in Number of Components

The last source of potential dissimilarity is the difference in the number of identical components in each of the compared existing and new BOM trees. Since the proposed reconciliation method we are concerned only with comparing BOMs trees, this type of dissimilarity is not considered initially. Such dissimilarity is to be taken into consideration if for a certain two pairwise matchings, between a new BOM tree and two existing ones, the total reconciliation cost given by Equation 2.4 was equal. In that case, difference in numbers of components is applied as an additional criterion to decide which of the two matching existing trees is more similar to the new one. The difference in number of components is given by the sum of absolute differences in numbers of each component that exist in both trees. The number to be considered for a certain component in a certain tree is given by the multiplication of the numbers of every ancestor node to that component and the number of the component itself.

For instance, the difference in number of components in the two trees, *T1* and *T2*, shown in Figure 2.5 is given by the sum of absolute differences in number of components *B*, *D* and *E*. For tree *T1*, there are quantity 2 from *B*, 2x3 from *D* and 2x2 from *E*. As for the tree *T2*, there are quantity 3 from *B*, 1x3 from *D* and 1x3 from *E*. Thus, the total difference in number of components should be 2 for difference in *B* plus 2 for difference in *D* plus 1 for difference in *E*, which gives a total difference of 5. With regard to subsequent process planning applications, based on retrieved matched BOM, the existence of multiple of identical components or subassemblies does not change the process plan (i.e. type of processes, their sequence and precedence relationships). Multiple identical components mean repeating a process several times accordingly and would generate additional assembly cost. However, this does not impact the comparison between Bill of Material Trees.

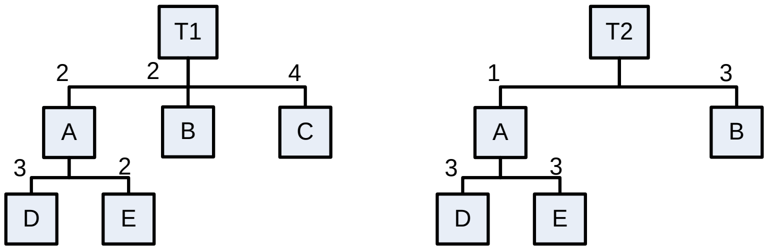


**Figure 2.5:** Calculating difference in number of components for two BOM trees *T1* and *T2*

### 2.5.5 Summary of Proposed Design Retrieval method using BOM Trees Matching

The proposed BOM trees matching method based on tree reconciliation is summarized in the following steps where the steps from 1 to 5 are to be repeated for each existing BOM tree before moving to step 6:

| | |
|---|---|
| **Step 1:** | Build a cladogram for the existing BOM tree |
| **Step 2:** | Build binary cladogram for the new BOM with extra items in the existing tree as pseudo items |
| **Step 3:** | Calculate duplication and loss events using NOTUNG 2.6 |
| **Step 4:** | Find extra items in the new BOM tree |
| **Step 5:** | Calculate total reconciliation cost using Equation 2.4 |
| **Step 6:** | If more than one existing BOM has the same total reconciliation cost with the new BOM, the best matching tree is the one with the least difference in numbers of components |

## 2.6 Case Study

A case study involving six different designs for chemical processing centrifugal pumps adopted from Liquiflo™ catalogues (http://www.liquiflo.com) is used for demonstrating the proposed BOM trees matching method based on tree reconciliation algorithm. Figure 2.6 shows a schematic drawing for one of the pump designs (pump 5). It is assumed that one of the six designs represents the new centrifugal pump while the other five are existing ones. Thus, it is required to find the existing pump design that is most similar to the new one. A BOM tree for each pump design (Figure 2.7) was constructed from labeled catalogue drawings. The main components only were considered excluding standard components such as bolts, nuts, screws, washers, plugs, pins, and keys in order to keep the size manageable for demonstration purposes. Accordingly, the total number of involved components is 23 and the number of components, for each pump design, ranges between 8 and 12 different components. BOM tree of pump 1, Centry 620 (Single Mechanical Seal), is assumed to be the new pump BOM tree, while the remaining five BOM trees are assumed to be existing trees in the manufacturer database.

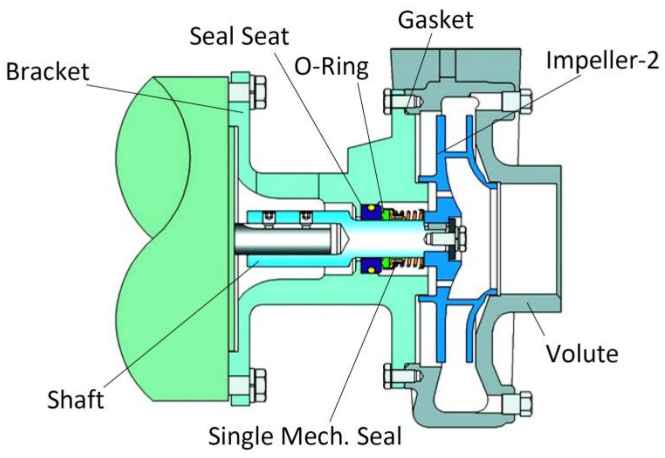


**Figure 2.6**: Pump 5: Centry 621 - Single Mechanical Seal (http://www.liquiflo.com)

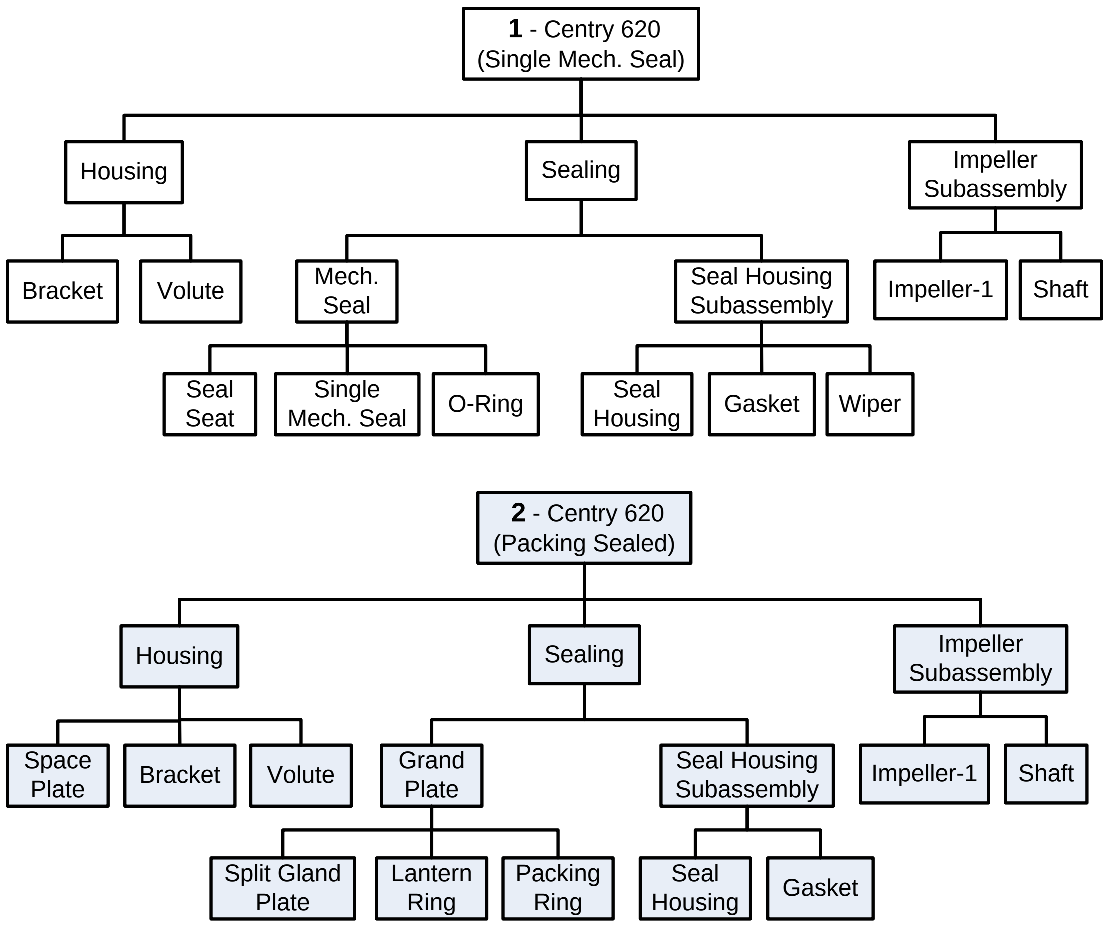


**Figure 2.7**: BOM trees for the six Liquiflo centrifugal pumps. BOM tree of pump 1, Centry 620 (Single Mech. Seal), is the new pump and other BOM trees are for existing pumps

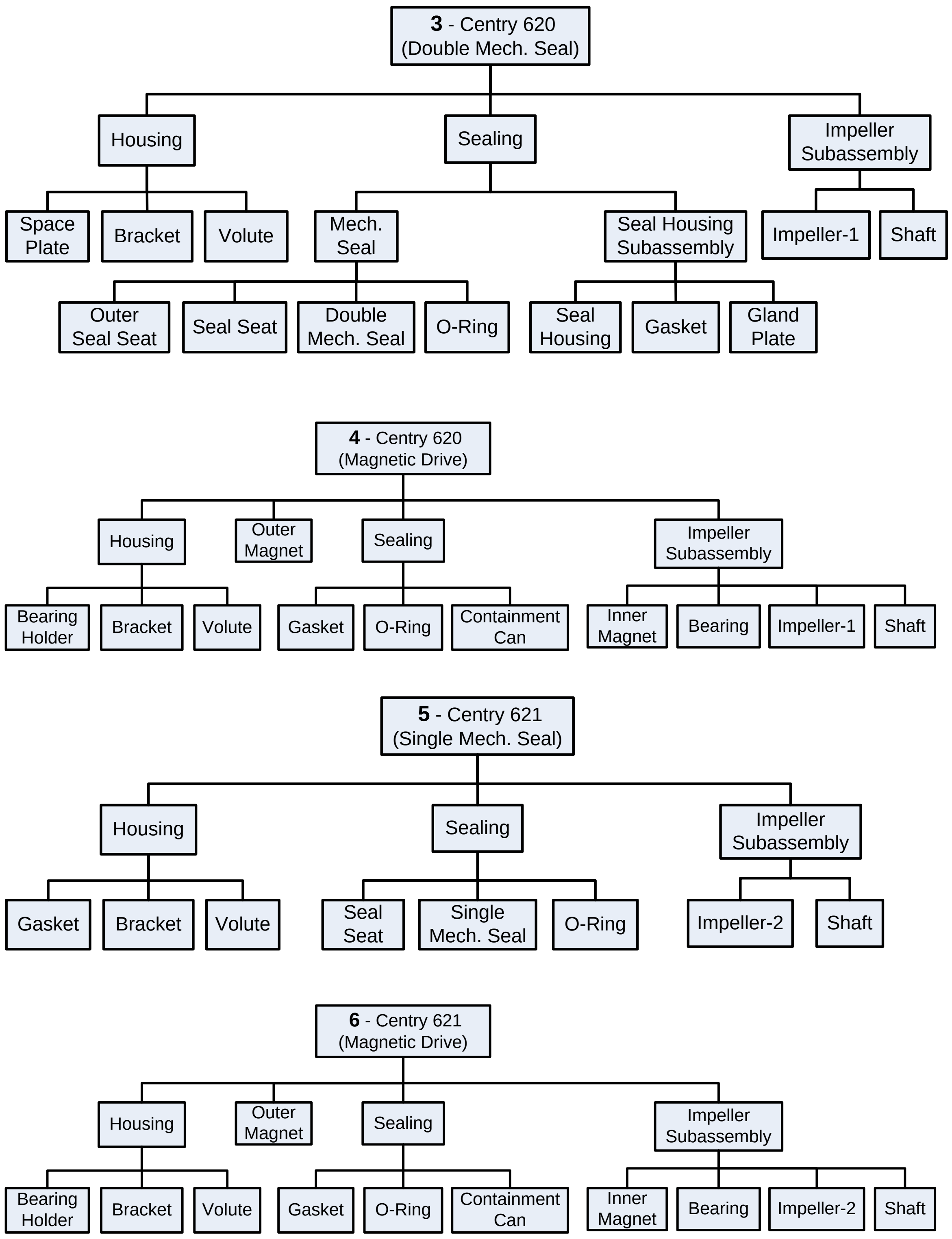
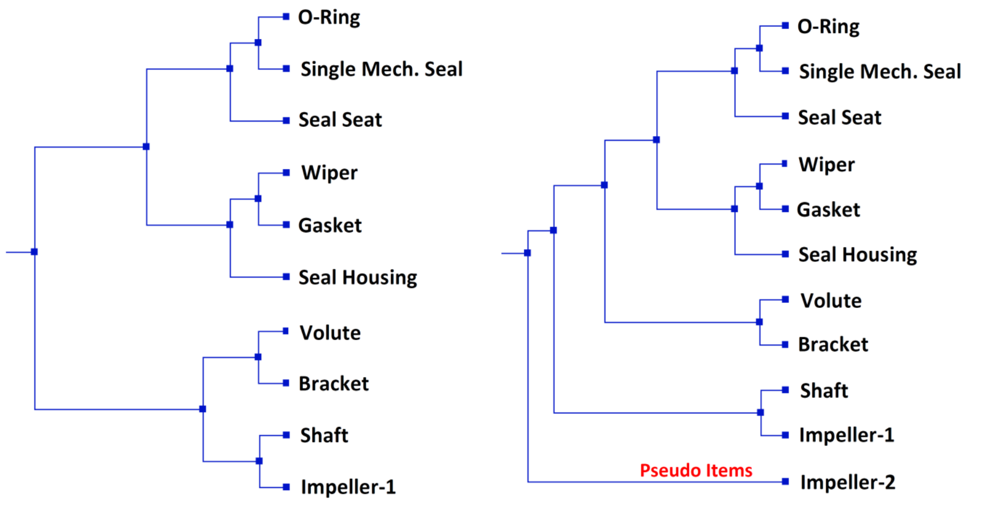


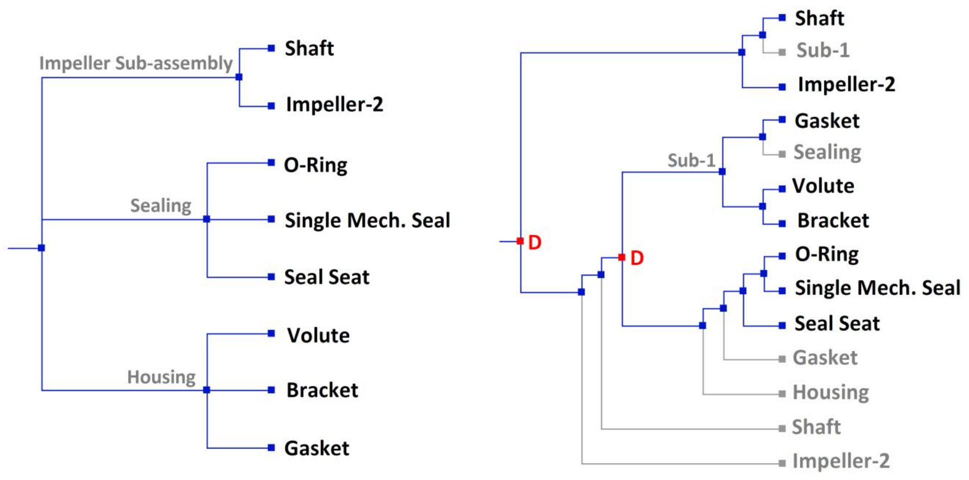
**Figure 2.7**: BOM trees for the six Liquiflo centrifugal pumps. BOM tree of pump 1, Centry 620 (Single Mech. Seal), is the new pump and the other BOM trees are for existing pumps study (Cont.)

As mentioned in Section 2.4, the BOM tree of the new pump should be converted first into a binary tree. A modified version of the binary tree is then generated for each of the existing five products BOM trees. Each modified tree includes an additional branch for pseudo components that exist in a given existing tree but not in the original new tree. For instance, Figure 2.8 shows the modified version corresponding to pump 5, where the component impeller-2, which does not exist in pump 5 is added as a pseudo component.  Figure 2.9 shows the BOM tree for pump 5, Centry 621 (Single Mechanical Seal), before reconciliation and the corresponding reconstructed tree after reconciliation. Duplication events are identified by capital letter D, while loss events are identified by pale branches. Duplication and loss events represent the differences between an existing product structure (BOM) and that of the new one (i.e. how an existing BOM tree should look like in order to match the new BOM tree).



**Original binary BOM tree** **Modified BOM tree (impeller-2 is added)**

**Figure 2.8**: Binary BOM tree for pump 1 before and after adding the pseudo items



**BOM tree for pump 5** **Reconstructed BOM tree for pump 5**

**Figure 2.9**: BOM tree for pump 5 before and after reconciliation with BOM tree of pump 1

Reconciliation costs (Equation 2.4) obtained for all pairwise matching between BOM tree of pump 1 and the other five trees, as well as the corresponding matching rank of each existing BOM tree are given in Table 1.1. The default unit cost used in NOTUNG software for duplication and loss events are applied ($C_D$ = 1.5 and $C_L$ = 1). A unit cost of 1 is assumed to be a reasonable estimate for the components term ($C_C$ = 1). Results of test runs have shown that the proposed method is not sensitive to the values assigned to these unit costs. Accordingly, BOM tree of pump 5 was found to be the best match for the BOM tree of pump 1; as it is the one corresponding to the least total reconciliation cost.

**Table 2.1:** BOM trees matching results

| BOM Tree | Duplication Events | Loss Events | Extra Components | Total Reconciliation Cost | Matching Rank |
|---|---|---|---|---|---|
| **Pump 2** | **2** | 10 | 4 | 17 | **2** |
| **Pump 3** | **3** | 16 | 3 | 23.5 | **3** |
| **Pump 4** | **3** | 18 | 4 | 26.5 | **4** |
| **Pump 5** | **2** | 6 | 3 | 12 | **1** |
| **Pump 6** | **3** | 20 | 5 | 29.5 | **5** |

It took less than 1 second on a PC of 2.8 GHz Quad Core processor and 4 GB RAM to perform the reconciliation step, which is the core of the proposed method (step 3). For a small dataset such as the one considered in this example, the results could have been obtained without a computerized algorithm. However, for cases where the number of existing BOM trees and the average number of components per tree are in the tens or hundreds, which is typical of many products, an efficient computerized tool must be used. The proposed trees reconciliation algorithm is scalable for larger size problems with numerous components and multiple hierarchical levels.

Using the proposed method based on tree reconciliation, rapid identification of similar current or existing designs that best match a new one in terms of contents and assembly structure could be efficiently realized. Once the most similar product design has been identified and retrieved, all related information can be accessed and modified by the designers and planners. For example, an assembly process plans, for a given new design, could be generated by making the necessary modifications on the assembly process plan corresponding to the best matching existing design, which is known in process planning literature as variant process planning (ElMaraghy 1993). The same concept applies to modifying CAD models, robotic assembly programs, tooling design, and the like.

## 2.7 Comparison with Results of Existing BOM Matching Techniques

The proposed BOM trees matching method based on tree reconciliation can be further verified by comparing the BOM trees matching results with those of the existing methods, albeit very limited. Of all techniques introduced for matching BOM trees, only shih (Shih 2011) has reported examples including more than one existing BOM tree. One of those examples presented by Shih to demonstrate his linear algebra based algorithm (Orthogonal Procrustes) consists of one new BOM and five existing BOMs (hypothetical dataset). The entire six BOM trees are shown in Figure 2.10, where the numbers found above a specific item (sub-assembly or component) represents the required number of that component.

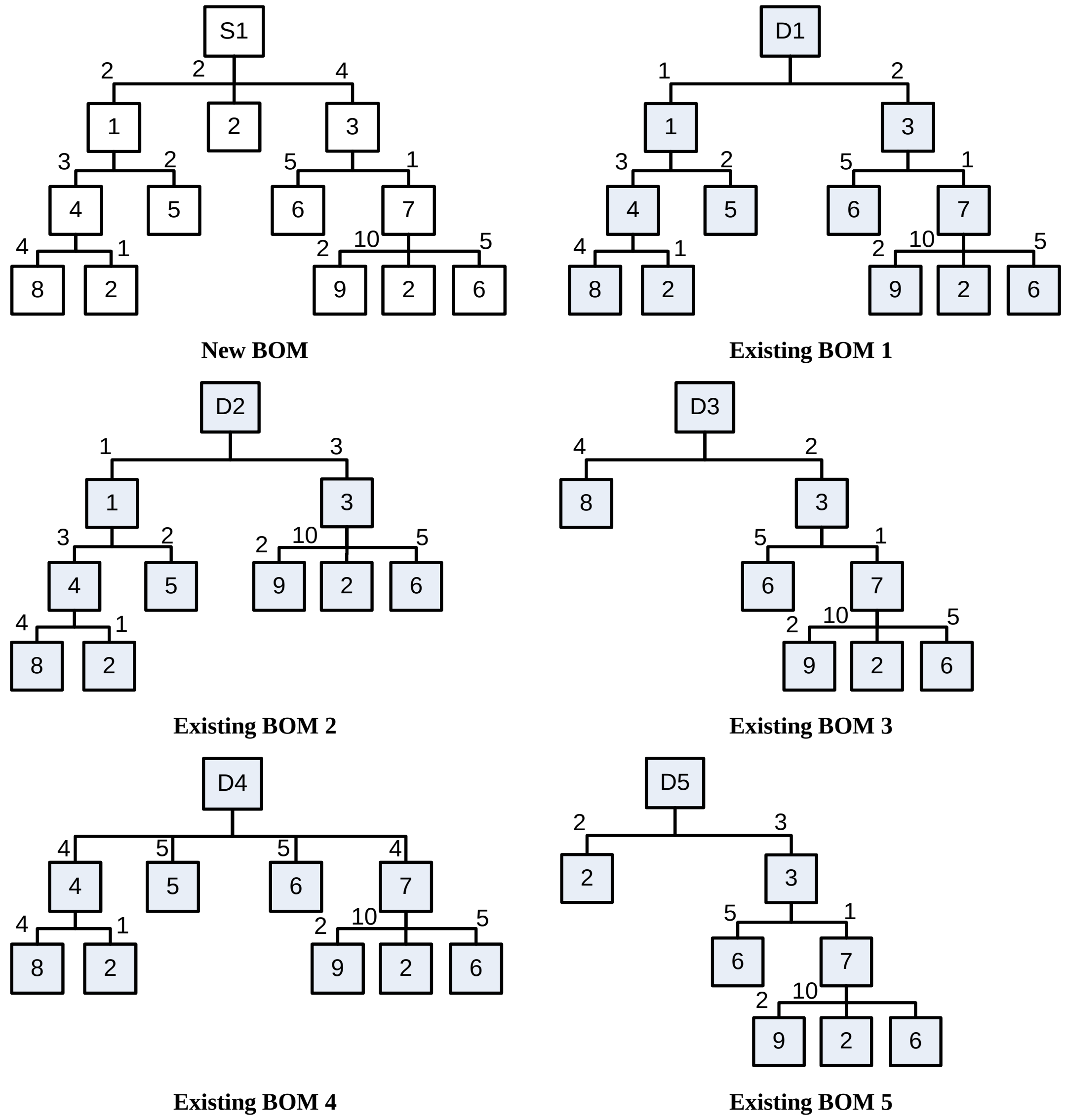


**Figure 2.10**: BOM trees for the example studied in Shih (reproduced from (Shih 2011))

The results obtained by Shih showed that the best matching BOM tree to the new one (S1) is existing BOM 1 (D1). When the proposed tree reconciliation matching method is applied to this example, different results were obtained, with D4 found to be the best matching tree to S1 instead of D1. According to the proposed method, both BOM trees D1 and D4 do have the same total reconciliation cost when matched with S1, however D4 is a better match for S1 as it has less difference in number of components. Such difference in results between the proposed method and that of Shih might be attributed to the different treatment of intermediate nodes by the two methods. Shih’s method, as well as other existing methods, seeks strict matching that takes into consideration intermediate nodes, while the presented method does not. For instance, Shih’s algorithm considers the existence of items (sub-assemblies) 1 and 3 in the new BOM tree (S1), and their absence from existing BOM 4 (D4) a dissimilarity, while the proposed method does not, because all sub-elements of both sub-assemblies, 1 and 3, do exist in both trees.

Explicitly considering intermediate nodes adds more unnecessary computational effort to a BOM matching algorithm and may lead to misleading results. For instance, as shown in Shih’S example in Figure 2.10, trees D2 and D4, each of them has the components 9, 2, and 6 together in the same sub-assembly, but under different names (sub-assembly 3 in D2 and sub-assembly 7 in D4). These mismatching names would overestimate the dissimilarity score if both trees were compared to each other using Shih’s algorithm. However, ignoring the name of the sub-assemblies (intermediate nodes) and only considering the fact that those three components (9, 2, and 6) exist together in the same sub-assembly, will lead to more reliable matching results.

The results of the proposed method for all pairwise tree reconciliations do not show any duplication or loss events, as none of the topologies of the five BOM trees actually conflicts with that of the new BOM tree. In other words, all the existing BOM trees can be considered as homomorphic (structure-preserving) sub-trees within the new BOM tree. Hence, dissimilarities arise only due to the extra components in the new BOM tree or the differences in the number of components. Table 2.2 summarizes the results of the proposed tree reconciliation-based matching method and Shih’s results.

**Table 2.2**: Comparing BOM matching results using the proposed method and Shih’s (Shih 2011)

| **BOM Tree** | **Duplication** | **Losses** | **Components** | **Total Reconciliation Cost** | **Number of Components** | **Matching Rank** | **Matching Rank by Shih (Shih 2011)** |
|---|---|---|---|---|---|---|---|
| **D1** | 0 | 0 | 1 | 1 | 51 | **2** | **1** |
| **D2** | 0 | 0 | 2 | 2 | 34 | **3** | **2** |
| **D3** | 0 | 0 | 3 | 3 | 59 | **5** | **3** |
| **D4** | 0 | 0 | 1 | 1 | 11 | **1** | **4** |
| **D5** | 0 | 0 | 3 | 3 | 21 | **4** | **5** |

The efficiency of the proposed BOM matching method based on tree reconciliation is well demonstrated through this example. The proposed method is capable of realizing the absence of conflict in topologies between the new BOM tree and all the existing BOM trees. Hence, the trees matching algorithm in this example is reduced to simple pairwise components and quantities comparison.

The computational complexity of Shih's linear algebra based algorithm is not reported; however, it is expected to be computationally intense due to using several matrix multiplication operations. Generally, multiplying two matrices is of a cubic complexity. The kernel of the tree reconciliation algorithm of NOTUNG is the duplication identification routine which is of a quadratic complexity. Therefore, the proposed method should be less computationally complex, which is an important criterion for large size datasets.

## 2.8 Summary

Well-developed tree reconciliation techniques were proposed in this chapter as the basis for a new method for matching Bill of Materials (BOM) trees for the purpose of retrieving similar legacy product designs. This is a "classification" type of application for knowledge discovery. Unlike Group Technology (GT) coding systems, and most other conventional design retrieval methods, which are mainly concerned with retrieval of individual parts based on similarity of geometric features, BOM matching addresses the assembly level, where similarity between whole products is sought based on their content (product components) and assembly structure.

A case study of six chemical processing centrifugal pumps was used to demonstrate the use of the developed method. The best matching existing BOM to the new BOM, the one with the least total reconciliation cost, was successfully obtained. Once the closest matching BOM has been identified and consequently the most similar product design, much of the useful data previously generated for such a design can be easily modified and used for the new product design with obvious productivity benefits. These include CAD design models, assembly process plans, assembly fixtures and tools, programs for assembly machinery and robots and supply chain information.

For further analysis and verification, the results of the proposed method were also compared with those found in the relevant most recent publications. Different matching outcomes are identified and are attributed to the different treatment of intermediate nodes by the two matching methods. This novel and straightforward BOM matching method, compared to existing methods, is less computationally complex as discussed in the comparison in section 2.7. The developed design retrieval method should result in considerable savings in product development time and effort; it is undoubtedly better to start from something existing than to start from scratch.

# CHAPTER 3
# PRODUCT FAMILY FORMATION BASED ON SIMILARITY OF PRODUCTS STRUCTURE

## 3.1 Introduction

The second product development application addressed in this research is product family formation. It is important for manufacturing system planners to identify sets of product families that shares significant manufacturing resources and setups to maximize system utilization and productivity and decrease system complexity. Many grouping methods have been developed for forming product families (Galan et al. 2007; Eguia et al. 2011; Pattanaik and Kumar 2011; Goyal et al. 2013) utilizing similarity measures such as components commonality, reusability, operations sequence as well as other measures. None of the existing grouping methods takes into consideration similarity in products structure which is important for assembly systems. It is not uncommon to find products which share many common components but with a considerably different assembly structure. For example, a group of water kettles might be sharing the same power switch but at different position in each product.

A new product family formation method based on a novel Bill of Material (BOM) trees similarity measure, which addresses both components commonality and hierarchical structure of products is proposed. The method is a clustering type of knowledge discovery that demonstrates how BOM trees matching used in Chapter 2 for product design retrieval could be extended for other useful design and manufacturing applications.

Due to some limitation of the tree reconciliation-based method presented in Chapter 2, to be discussed later in the review section, a new BOM matching method is proposed. The new method is based the Robinson-Foulds (RF) distance (Robinson and Foulds 1981) to be utilized in Chapter 5 for generating master assembly trees, in which an enhanced and modified version (RF-BOM) is developed. A novel Integer Programming model is formulated and implemented to formally define and calculate the new measure of distance between any given pair of BOM trees. Average linkage hierarchical clustering is then used to build clustering trees based on the pairwise similarity distances obtained from BOM matching.

## 3.2 Literature Review

### 3.2.1 Product Family Formation

Much research work has been carried out to study and address product variety from various perspectives (Rajagopalan and Xia 2012; ElMaraghy et al. 2013; Takenaka et al. 2013; ElMaraghy and ElMaraghy 2014; Patel and Jayaram 2014). The notion of grouping products into families to capitalize on similarity within a class of products is considered a pre-requisite for success in managing variety (ElMaraghy and ElMaraghy 2014).

State-of-the-art product family formation methods primarily use average linkage hierarchical clustering (Seifoddini and Wolfe 1986) to cluster products into a binary rooted tree called dendrogram. However, the main difference between them is the similarity measures on which the clustering is based. Abdi and Labib (2004) used operational similarities between products calculated using Jaccard's similarity coefficient (McAuley 1972) which is a commonly used similarity coefficient for part-cell formation in cellular manufacturing. Goyal et al. (2013) proposed a similarity coefficient based on operations sequence. Galan et al. (Galan et al. 2007) used five similarity coefficients; modularity, commonality, compatibility, reusability and demand. Analytic Hierarchy Process (AHP) (Saaty 1986) is used as a weighing method to aggregate the five similarity coefficients into one single coefficient. Eguia et al. (2011) addressed the product family formation for disassembly systems using a similarity coefficient using the following information: a) types and quantities of the products to disassemble within a certain time horizon, b) existing reconfigurable machine tools and available modules library, c) operations and processing times required to disassemble each product type, and d) machines and modules required for each disassembly task.

Considering alternative process plans, Rakesh et al. (2010) proposed a modified average linkage hierarchical clustering algorithm based on Jaccard's similarity coefficient. Abdi (2012) proposed a conceptual framework for product family formation using Analytical Network Process (ANP) (Saaty 1996), which is a developed form of the AHP that allows more interrelationships among decision elements. Abdi's framework incorporates six major clustering criteria mainly related to manufacturing operations and market requirements, where each major criterion is further broken down into more elements that affect product family formation and selection. Pattanik and Kumar (2011) implemented a multi-objective (bi-criterion) Genetic Algorithm (Deb et al. 2002) to cluster products based on pre-defined number of clusters using two similarity coefficients - components commonality and demand.

Whenever components commonality is used as a similarity measure for an assembly application, the similarity in product assembly structure should also be taken into consideration, particularly when the structures of involved product variants are different. Products sharing many common components may have considerably different product structures; hence, grouping such products into one family would not be useful. Commonality measures used in product family design/redesign, such as the Commonality versus Diversity Index (CDI)  (Alizon et al. 2009), the Comprehensive Metric for Commonality (CMC) (Thevenot and Simpson 2007) and Total Commonality Index (TCI) (Blecker and Abdelkafi 2007) do not capture the similarity in product structure. A BOM tree is a conventional and widely accepted form for representing a given product structure. Properly measuring the distance between any given pair of BOM trees should provide a reliable aggregate measure for commonality of components as well as the similarity of the assembly structure of corresponding products.

### 3.2.2 Matching BOM Trees

A thorough literature review for existing BOM trees matching methods was presented in Chapter 2 (Section 2.2) and it has been mentioned that existing methods are taking into consideration the strict matching of node (sub-assembly) names which adds unnecessary computational effort to a BOM matching algorithm and may lead to misleading results (i.e. over-estimated distances). BOM trees matching based on tree reconciliation avoided such a shortcoming and is computationally efficient; however, the drawback of the tree reconciliation-based method is that it requires at least one of any pair of BOM trees to be a binary tree. A binary tree is a tree where each of its nodes has exactly two branches. Thus, there is a need to convert non-binary BOM trees, which is quite common, into binary trees in order to apply the algorithm. Such a conversion may result in different matching results for different non-binary to binary tree conversions. This is not a significant limitation in a design retrieval application, in which only the new non-binary BOM tree is converted to a binary tree while existing ones are not. However, for a product family formation algorithm, where every pair of trees is compared and many binary to non-binary conversions would take place, it would lead to unacceptable inconsistencies.

## 3.3 Research Scope and Assumptions

The scope of this chapter is to develop a method for grouping several different variants (e.g. models or types) of a given product into number of groups or families based on similarity in BOM. The proposed generic grouping method is not specific to certain manufacturing systems, such as cellular or reconfigurable manufacturing systems. The targeted application is assembly where product hierarchical structure plays an important role. All BOM matching assumptions used in Chapter 2 for product design retrieval also applies in this chapter. It is assumed that versions and variants of each component or sub-assembly are first grouped under the same label or identification code. Besides, it is assumed that a consistent methodology is followed in constructing all existing BOM trees. It is also assumed that same language and consistent terminology is used to describe elements of involved BOMs.

## 3.4 Proposed Product Family Formation Method

Robinson-Foulds (RF) distance is a commonly used measure for comparing phylogenetic trees that are not necessarily binary. The proposed product family formation method is based on an enhanced and customized version of Robinson-Foulds distance (RF-BOM) that provides a more accurate tree dissimilarity measure. An Integer Programming model is formulated to mathematically define, as well as to calculate the modified Robinson-Foulds distance between any given pair of BOM trees. Average linkage hierarchical clustering is then used to form product families based on the pairwise dissimilarity distances obtained through BOM trees matching. A flowchart summarizing the steps of the proposed product family formation method is shown in Figure 3.1.

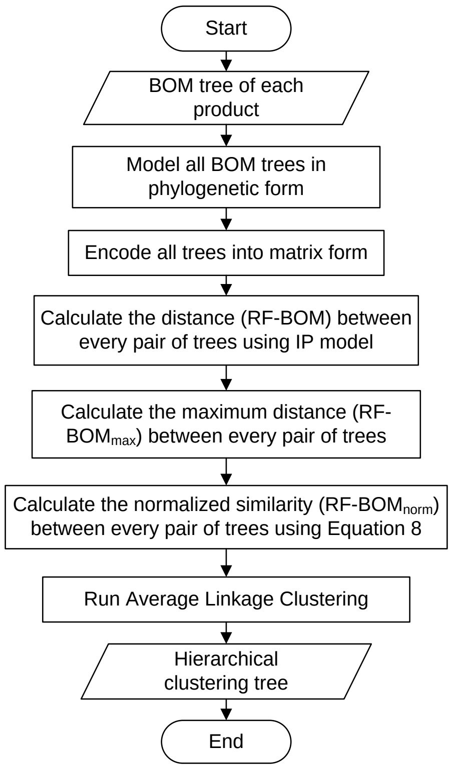

Start
BOM tree of each product
Model all BOM trees in phylogenetic form
Encode all trees into matrix form
Calculate the distance (RF-BOM) between every pair of trees using IP model
Calculate the maximum distance (RF-BOM$_{max}$) between every pair of trees
Calculate the normalized similarity (RF-BOM$_{norm}$) between every pair of trees using Equation 8
Run Average Linkage Clustering
Hierarchical clustering tree
End

**Figure 3.1**: A flowchart summarizing the proposed product family formation method

### 3.4.1 Modeling a BOM Tree as a Phylogenetic Tree

Figure 3.2 demonstrates how a conventional BOM tree could be represented as a phylogenetic tree (cladogram), where positive integer numbers are used to encode the name of any given component. Arbitrarily ordered numbers are also used in the phylogenetic tree form to identify nodes. Unlike the BOM modeling scheme used in Chapter 2, general non-binary trees in this chapter are not restricted by any means.

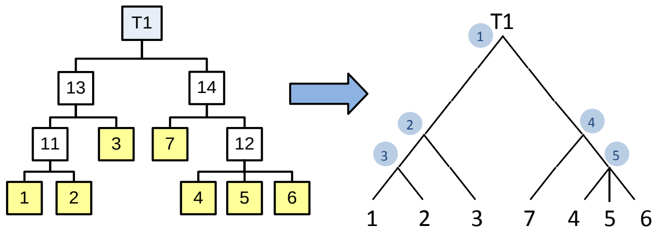


**Figure 3.2**: Modelling a BOM tree as a phylogenetic tree

In the conventional BOM tree form, the required number of any given sub-assembly or component is provided on the tree. A new scheme is proposed in order to make it more straightforward for comparing differences in numbers of sub-assemblies or components, as a dissimilarity aspect. Accordingly, the total numbers required of components only are provided on the tree. The total number of a certain component is given by the multiplication of the number required of that component and the number required of every sub-assembly which includes that component. This scheme is consistent with the notion introduced before in Chapter 2 of only dealing with components and considering sub-assemblies as no more than groups of components.

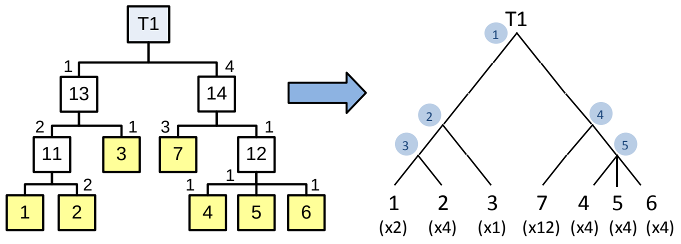


**Figure 3.3**: Modelling a BOM tree considering components quantities

Figure 3.3 shows a modified case of the example shown in Figure 3.2, in which more than one of some components and sub-assemblies are required, and it demonstrates how the phylogenetic-based tree form is modified accordingly. For instance, four of component 2 are required because two of sub-assembly 11 and two of component 2 are required. The benefit of using such a scheme is illustrated in the case of comparing two BOM trees which have the same sub-assembly with the same number of replications but with different numbers of required lower-subassemblies and/or components. The resulting distance from matching this sub-assembly using any BOM trees matching method would inaccurately be zero. The proposed BOM tree modeling scheme avoids this inaccuracy.

### 3.4.2 Modified Robinson-Foulds Distance for BOM Trees (RF-BOM)

Robinson-Foulds (RF) distance, described in detail in Chapter 4, in is a normalized count of the nodes (i.e. groups of leaves) that exist in one tree, but not the other. The main limitation of such a measure for the proposed BOM trees matching application is that it does not accurately assess the degree by which a tree node (sub-assembly) in a given tree is absent or different from the other one. Furthermore, the RF distance does not capture dissimilarity in numbers of the same tree node or leaf. In the example in Figure 3.4, the RF distance between *T1* and *T2* is given by 1/2 (3+3) = 3, as there are three nodes in *T1* that do not exist in *T2* (nodes 1, 4 and 5) and there are three nodes in *T2* as well that do not exist in T1 (nodes 1, 4 and 5). The distance between *T1* and *T3* is also given by 1/2 (3+3) = 3, as there are three nodes in *T1* that do not exist in *T3* and there are three nodes in *T2* as well that do not exist in *T1*.

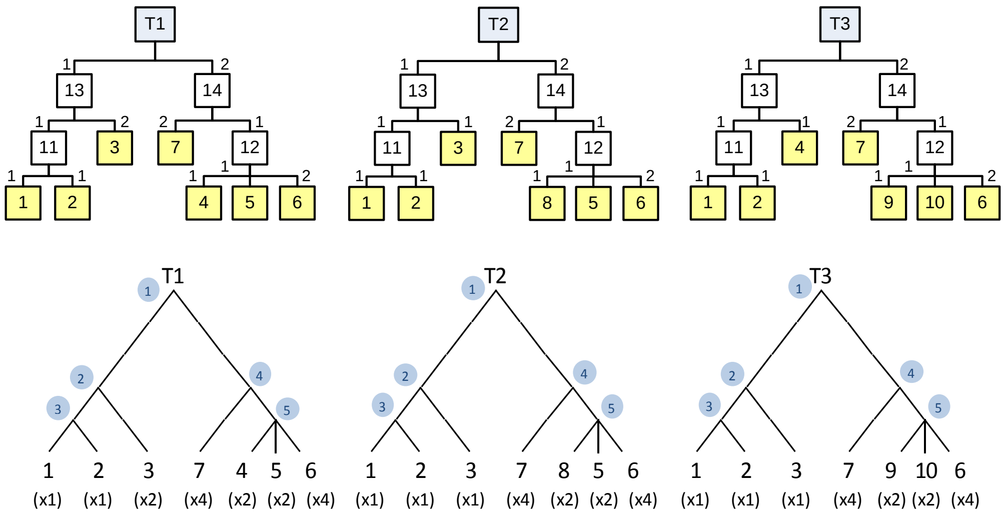


**Figure 3.4**: An example for BOM trees in traditional and phylogenetic form

The original RF distance gives the same result when comparing *T1* to each of *T2* and *T3*, while in fact the dissimilarity between *T1* and *T2* should be more than the dissimilarity between *T1* and *T3*. By looking at the pair of trees *T1* and *T3*, Tree 1 has 2 component that do not exist in Tree3 (components 4 and 5), and *T1* has 2 components that do not exist in T3 (components 9 and 10). While for the pair of trees *T1* and T2, there is only one component in *T1* that does not exist in *T2* (component 4); and there is one component also in *T2* that does not exist in *T1* (component 8). Hence, the original RF distance does not accurately capture the difference between a given pair of BOM trees.

The proposed modified version of RF distance accounts for the degree by which a given node (sub-assembly) in one tree is different from the corresponding node in another tree. This is achieved by performing bi-directional mapping when comparing a given pair of BOM trees *T1* and *T2*. Accordingly, every node in *T1* is mapped to the most similar node in *T2*, and every node in *T2* is mapped as well to the most similar node in *T1*. The distance between any node *d1* in one tree which is mapped to another node *d2* in another tree is given by the number of components that exist in *d1* but not *d2*. Furthermore, if any mismatched component in *d1* does not directly belong to *d1*, the distance is divided by the number of hierarchical levels by which the node and the mismatching component are separated. This is to reduce the impact of mismatching components which belong to lower-level subassemblies on higher-level ones.

The total modified RF distance is thus given by the sum of all distances between all pairs of mapped nodes in both directions. Accordingly, the distance between *T1* and *T2*, in Figure 3.6, could be calculated as shown in Tables 3.1 and 3.2. The sum of distances obtained from mapping *T1* nodes to *T2* nodes is 1 5/6 and the sum of distances obtained from mapping *T2* nodes to *T1* nodes is 1 5/6. Thus the total distance between the two trees is given by 1 5/6 + 1 5/6 = 3 2/3. Following the same procedure, a larger distance is calculated between *T1* and *T3*, which turns out to be 3 2/3 +3 2/3 = 7 1/3.

**Table 3.1**: Mapping T1 nodes to T2 nodes and resulting distances

| **Node in *T1*** | **Most similar node in *T2*** | **Components in *T1* absent from *T2*** | **Relative Level** | **Distance** |
|---|---|---|---|---|
| 1 | 1 | 4 | 3 | 1/3 |
| 2 | 2 | - | - | 0 |
| 3 | 3 | - | - | 0 |
| 4 | 4 | 4 | 2 | 1/2 |
| 5 | 5 | 4 | 1 | 1/1 |
| **Total** | | | | 1 5/6 |

**Table 3.2**: Mapping T2 nodes to T1 nodes and resulting distances

| **Node in *T2*** | **Most similar node in *T1*** | **Components in *T2* absent from *T1*** | **Relative level** | **Distance** |
|---|---|---|---|---|
| 1 | 1 | 8 | 3 | 1/3 |
| 2 | 2 | - | - | 0 |
| 3 | 3 | - | - | 0 |
| 4 | 4 | 8 | 2 | 1/2 |
| 5 | 5 | 8 | 1 | 1/1 |
| **Total** | | | | 1 5/6 |

In order to account for dissimilarity in required number of components, the dissimilarity distance due to a given component existing in one tree but not the other is multiplied by the number required of that component. Furthermore, any component that exists in one tree with a higher number than in the other one; the resulting dissimilarity distance is given by the additional number required of that component divided by its relative hierarchical level, with respect to its node. As shown in Figure 3.4, the total number of some components that are shared among *T1* and *T2* is not the same (component 3), and more than one of the components that are not shared are required (components 4 and 8). In this case, the distance between *T1* and *T2* should be calculated as shown in Tables 3.3 and 3.4 which turns out to be 4 2/3 + 3 2/3 = 8 1/3. Henceforth, the modified RF distance will be referred to as RF-BOM and its mathematical description is introduced in Section 4.4.

**Table 3.3**: Mapping T1 nodes to T2 nodes and resulting distances considering numbers required

| **Node in *T1*** | **Most similar node in *T2*** | **Comp. in *T1* absent from *T2*** | **Comp. with higher quantity in *T1*** | **Rel. level** | **Diff. in quant.** | **Distance** |
|---|---|---|---|---|---|---|
| 1 | 1 | 4 | - | 3 | 2 | 2/3 |
| 2 | 2 | - | 3 | 1 | 1 | 1/1 |
| 3 | 3 | - | - | - | - | 0 |
| 4 | 4 | 4 | - | 2 | 2 | 2/2 |
| 5 | 5 | 4 | - | 1 | 2 | 2/1 |
| **Total** | | | | | | 4 2/3 |

**Table 3.4**: Mapping T2 nodes to T1 nodes and resulting distances considering numbers needed

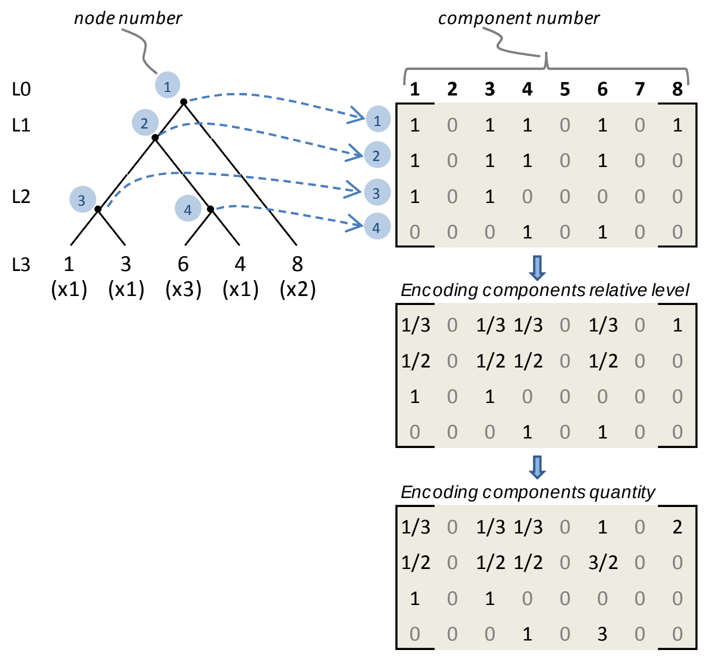
| **Node in *T2*** | **Most similar node in *T1*** | **Comp. in *T2* absent from *T1*** | **Comp. with higher quantity in *T2*** | **Rel. level** | **Diff. in quant.** | **Distance** |
|---|---|---|---|---|---|---|
| 1 | 1 | 8 | - | 3 | 2 | 2/3 |
| 2 | 2 | - | - | - | - | 0 |
| 3 | 3 | - | - | - | - | 0 |
| 4 | 4 | 8 | - | 2 | 2 | 2/2 |
| 5 | 5 | 8 | - | 1 | 2 | 2/1 |
| **Total** | | | | | | 3 2/3 |

### 3.4.3 Proposed BOM Trees Matching Integer Programming Model

#### *Trees Encoding*

A tree-to-matrix encoding scheme is proposed to represent any given BOM tree into a matrix form to be handled by the proposed model. As shown in Figure 3.5, a matrix of *m* rows and *n* columns is used to encode a BOM tree of *m* subassemblies (including the main assembly itself) and *n* components. Columns of the matrix represent components and rows represent nodes. Binary (0-1) values are used, as a first step, to represent the presence or absence of a given component in any given node. For instance, node number 2 in the tree in Figure 3.5 has components 1, 3, 6, and 4, thus, row number 2 in the corresponding encoded matrix will take the value 1 in the cells number 1, 3, 6, and 4, while all other cells will take a 0 value due to the absence of those components in node 2.

The encoding scheme is adjusted next to account for the relative hierarchical level by which a given component belongs to a given node. Accordingly, the value of each cell representing a component that belongs to a given node is divided by the number of hierarchical levels separating the component from the node. For example, cells representing components 1, 3, 6, and 4 in node 1 are divided by 3 (3 - 0), which is the difference between the hierarchical level of node 1 (level 0) and the hierarchical level of the components (level 3). The value of the cell representing component 8 in node 1 is not modified as component 8 directly belongs to node 1 since they are separated by only one level. The encoded matrix is complete after multiplying the value representing each component in each node by the required number. Accordingly, the value for cells representing component 6 is multiplied by 3 and the value representing component 8 is multiplied by 2.


node number
component number
L0
L1
L2
L3
1
3
6
4
8
(x1)
(x1)
(x3)
(x1)
(x2)
Encoding components relative level
Encoding components quantity

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Encoding components relative level

$$\begin{bmatrix} 1/3 & 0 & 1/3 & 1/3 & 0 & 1/3 & 0 & 1 \\ 1/2 & 0 & 1/2 & 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Encoding components quantity

$$\begin{bmatrix} 1/3 & 0 & 1/3 & 1/3 & 0 & 1 & 0 & 2 \\ 1/2 & 0 & 1/2 & 1/2 & 0 & 3/2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 3 & 0 & 0 \end{bmatrix}$$

**Figure 3.5**: Proposed tree-to-matrix encoding scheme

## ***Mathematical Formulation***

An Integer programming model for calculating the modified Robinson-Foulds distance (RF-BOM) is written and solved using LINGO optimization software package. The model input parameters are:

| | |
|---|---|
| $n$ | Total number of different components |
| $m1$ | Number of nodes (sub-assemblies) of tree 1 |
| $m2$ | Number of nodes (sub-assemblies) of tree 2 |
| $T1_{ij}$ | An element in the $i^{th}$ row and $j^{th}$ column of the encoded matrix representing tree 1 |

*$T2_{kj}$* An element in the $k^{th}$ row and $j^{th}$ column of the encoded matrix representing tree 2

And the decision variables are as follows:

| | |
|---|---|
| *$x1_{ik}$* | A binary (0-1) variable that takes 1 if node *i* in tree 1 is mapped to node *k* in tree 2. |
| *$x2_{ki}$* | A binary (0-1) variable that takes 1 if node *k* in tree 2 is mapped to node *i* in tree 1. |
| *$y1_{ik}$* | An auxiliary positive variable that represents distance from node *i* in tree 1 to node *k* in tree 2. |
| *$y2_{ki}$* | An auxiliary positive variable that represents distance from node *k* in tree 2 to node *i* in tree 1. |

An Integer Programming (IP) model with four sets of constraints is formulated, according to the listed parameters and decision variables, as follows:

$$\text{Minimize} \sum_{i=1}^{m1}\sum_{k=1}^{m2} y1_{ik} + \sum_{k=1}^{m2}\sum_{i=1}^{m1} y2_{ki} \qquad (3.1)$$

$$Subject\ to$$

$$y1_{ik} + M(1 - x1_{ik}) \geq \sum_{j=1|T1_{ij}\geq T2_{kj}}^{n} T1_{ij} - T2_{kj} \qquad \forall\, i = 1, \dots, m1 \quad \forall\, k = 1, \dots, m2 \qquad (3.2)$$

$$y2_{ki} + M(1 - x2_{ki}) \geq \sum_{j=1|T2_{kj}\geq T1_{ij}}^{n} T2_{kj} - T1_{ij} \qquad \forall\, k = 1, \dots, m2 \quad \forall\, i = 1, \dots, m1 \qquad (3.3)$$

$$\sum_{k=1}^{m2} x1_{ik} = 1 \qquad \forall\, i = 1, \dots, m1 \qquad (3.4)$$

$$\sum_{i=1}^{m1} x2_{ki} = 1 \qquad \forall\, k = 1, \dots, m2 \qquad (3.5)$$

The *y1*-decision variables are the distances from every node (sub-assembly) in tree 1 to every other node in tree 2, and the *y2*-variables are the distances from every node in tree 2 to every other node in tree 1. The model only assign a value to a *y1*-variable that measures the distance from a given node in tree 1 to only the most similar node in tree 2. The same also applies to the *y2*-variables. Hence, the objective function (Equation 3.1) provides the total sum of all *y1*- and *y2*-variables, which is the modified R-F distance (RF-BOM) between the two trees.

The first two sets of constraints (Equations 3.2 and 3.3) express the *y1*- and *y2*-variables, respectively. A *y1*-variable is the sum of the elementary subtraction operations between a row (node) in the encoded matrix representing tree1 and a row in the encoded matrix representing tree 2 (Equation 3.2).

Similarly, a *y2*-variable is the sum of the elementary subtraction operations between a row (node) in the encoded matrix representing tree 2 and a row in the encoded matrix representing tree 1 (Equation 3.3). Subtraction operations yielding positive values only are considered because it is only required to find components that exist in one tree but not the other or shared components that exist with larger numbers in one tree than the other. The binary *x1*- and *x2*-variables in the model, along with large constant, *M*, are used to ensure that the *y1*- and *y2*-variables are assigned zero values whenever the distance to be measured is between two nodes that are not mapped to each other. A rough safe estimate for *M* could be the sum of the values in the two encoded matrices. The other two sets of constraints (Equations 3.4 and 3.5) define such a mapping between the two trees. The third constraint (Equation 3.4) states that each node in tree 1 is mapped to only one node in tree 2, and the fourth constraint (Equation 3.5) states that each node in tree 2 is mapped to only one node in tree 1.

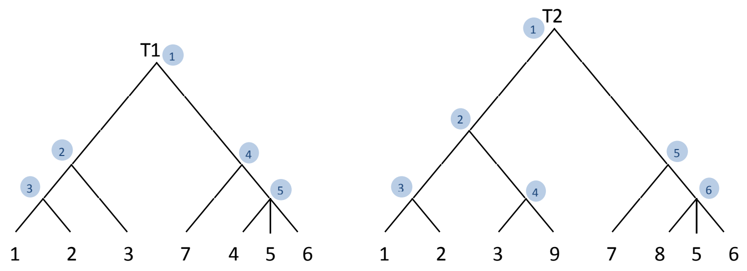
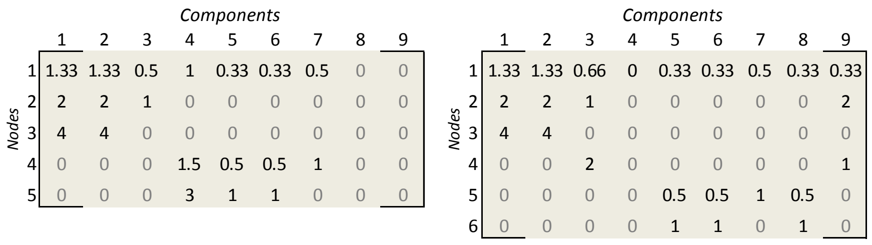


**Figure 3.6**: Two trees *T1* and *T2* with a RF-BOM distance of 11.82



| Nodes \ Components | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.33 | 1.33 | 0.5 | 1 | 0.33 | 0.33 | 0.5 | 0 | 0 |
| 2 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1.5 | 0.5 | 0.5 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 3 | 1 | 1 | 0 | 0 | 0 |

| Nodes \ Components | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.33 | 1.33 | 0.66 | 0 | 0.33 | 0.33 | 0.5 | 0.33 | 0.33 |
| 2 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 3 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0.5 | 0.5 | 1 | 0.5 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

**Figure 3.7**: Encoded matrices for *T1* and *T2*

For the two trees *T1* and *T2* shown in Figure 3.6, the corresponding encoded matrices are shown in Figure 3.7. The obtained *x1*- and *x2*-variables, representing the nodes which are mapped to each other, as well as the *y1*- and *y2*-values, representing the calculated distances between mapped nodes, are all shown Figure 3.8. Accordingly, the objective function representing the RF-BOM distance between the two trees is found to be 11.82 which is simply the total sum of the *y1*- and *y2*-matrices. For hierarchical clustering to be carried out, the convention is to build clustering trees using normalized similarity measures and not absolute dissimilarity measures as obtained by the proposed BOM trees matching model. Accordingly, a rough estimate for the maximum dissimilarity (RF-BOM$_{max}$) between any pair of BOM trees is assumed and then the similarity measure between that pair is given by

Equation 3.6. One possible rough estimate for maximum dissimilarity between two BOM trees is the total sum of values in the corresponding encoded matrices. This represents the case in which all nodes are completely mismatched. For instance, the maximum dissimilarity for the pair of BOM trees in Figure 3.6 is given by 26.82 + 28.64 = 55.46. Thus, the normalized similarity between that pair of BOM trees turns out to be (55.46 – 11.82)/55.46 = 0.787 or 78.7%. The maximum value for RF-BOM$_{norm}$ is 1 and its minimum value is 0.

$$\mathrm{RF-BOM_{norm}} = \frac{\mathrm{RF{-}BOM_{max}{-}RF{-}BOM}}{\mathrm{RF{-}BOM_{max}}} \qquad (3.6)$$

Nodes in T2

$$\begin{array}{c|cccccc} \textit{Nodes in T1} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

(a) *x1*-variables

Nodes in T1

$$\begin{array}{c|ccccc} \textit{Nodesin T2} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 \end{array}$$

(b) *x2*-variables

Nodes in T2

$$\begin{array}{c|cccccc} \textit{Nodes in T1} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1.5 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 3 \end{array}$$

(c) *y1*-variables

Nodes in T1

$$\begin{array}{c|ccccc} \textit{Nodesin T2} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0.82 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 2 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0.5 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 \end{array}$$

(d) *y2*-variables

**Figure 3.8**: Obtained values for different decision variables

The model was validated by comparing results obtained by the model with manually calculated results for small examples. The model is of a polynomial time complexity; the RF-BOM could in fact be calculated, for any given pair of BOM trees, by enumerating the distance between every node in one BOM tree with every node in the other tree.

## 3.4.4 Hierarchical Clustering

After calculating the normalized similarity between every pair of BOM trees for a given group of products, an agglomerative (bottom-up) hierarchical clustering algorithm, using an average-linkage method (Seifoddini and Wolfe 1986), is used to build the corresponding hierarchical clustering tree, also known as dendrogram. The input required by a hierarchical clustering algorithm is a symmetric square matrix representing similarity between every pair of products. Table 3.5 shows one example of a similarity matrix for a group of four products.

**Table 3.5** An example for a similarity matrix based on RF-BOM$_{norm}$

| **Product** | **A** | **B** | **C** | **D** |
|---|---|---|---|---|
| **A** | - | 0.20 | 0.75 | 0.20 |
| **B** | | - | 0.40 | 0.50 |
| **C** | | | - | 0.40 |
| **D** | | | | - |

The steps of a typical agglomerative hierarchical clustering algorithm are as follows:

1. Start with *N* product families, each initially representing a single product.
2. Search the similarity matrix for the closest pair of product families (i.e. the two families with the highest similarity). Denote those most similar families as *U* and *V*.
3. Merge *U* and *V* into a new family *W*. Update the similarity matrix by deleting the rows and columns corresponding to *U* and *V*, and adding a row and column representing the similarity between *W* and all other remaining families.
4. Repeat steps (2) and (3) a total of *N-1* times.

When products are merged into families during the algorithm, similarity between newly formed families and other families needs to be recalculated as mentioned in step 3. For average linkage clustering, the similarity $S_{pq}$ between any given pair of families *p* and *q* is re-formed according to Equation 3.7, where *i* and *j* are products in families *p* and *q* respectively, $S_{ij}$ is the similarity between product *i* and *j*, $N_p$ and $N_q$ are the number of products in families *p* and *q*. This simply means that the similarity between families is the average similarity between their pairs of products.

$$S_{pq} = \frac{\sum_{i \in p} \sum_{j \in q} S_{ij}}{N_p N_q} \quad (3.7)$$

The hierarchical clustering tree (dendrogram) obtained based on the RF-BOM similarity matrix given Table 3.5 is shown in Figure 3.9. According to this tree, at 75% similarity, the four products are clustered into three families: {A, C}, {B} and {D}; while at 50% similarity they are clustered into two families: {A, C}, {B, D}.

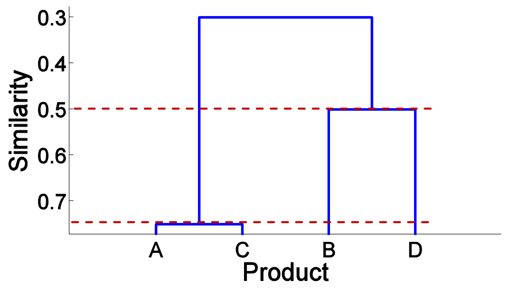


**Figure 3.9**: Obtained dendrogram based on the RF-BOM similarity matrix in Table 3.5

## 3.5 Case Study

The pumps case study used to demonstrate the BOM trees matching method based on tree reconciliation presented in Chapter 2 is used in this section to demonstrate the proposed product family formation method. That example involves six different designs for chemical processing centrifugal pumps, adopted from Liquiflo™ catalogues (http://www.liquiflo.com). It is assumed that the six variants need to be clustered into families of products. A BOM tree for each pump (Figure 2.7) was constructed from labeled company catalogue drawings.

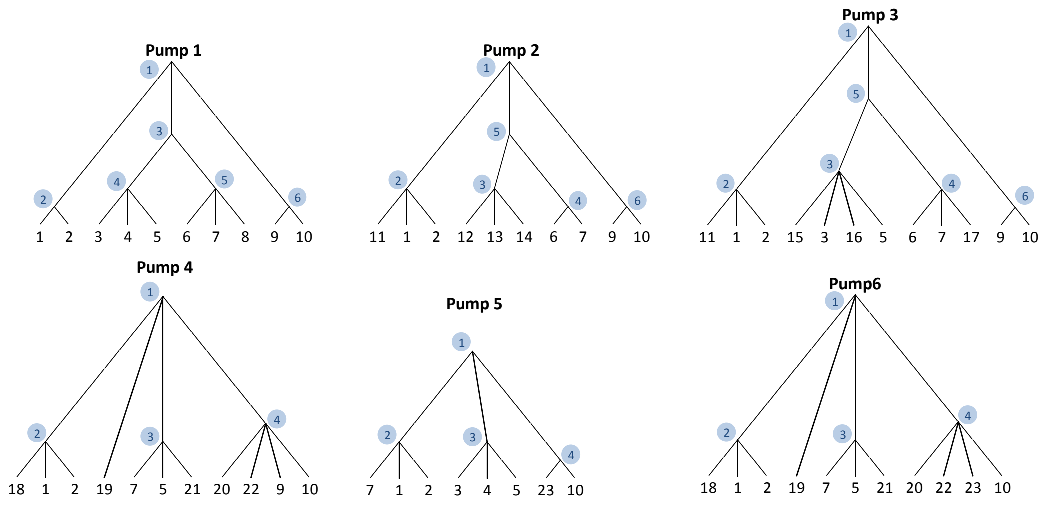


**Figure 3.10**: BOM trees in phylogenetic form for the six pumps

**Table 3.6**: Components names and the number assigned to each component

| Component name | Component number | Component name | Component number |
|---|---|---|---|
| Bracket | 1 | Lantern Ring | 13 |
| Volute | 2 | Packing Ring | 14 |
| Seal Seat | 3 | Outer Seal Seat | 15 |
| Single Mech. Seal | 4 | Double Mech. Seal | 16 |
| O-ring | 5 | Gland Plate | 17 |
| Seal Housing | 6 | Bearing Holder | 18 |
| Gasket | 7 | Outer Magnet | 19 |
| Wiper | 8 | Inner Magnet | 20 |
| Impeller-1 | 9 | Containment Can | 21 |
| Shaft | 10 | Bearing | 22 |
| Space Plate | 11 | Impeller-2 | 23 |
| Split Gland Plate | 12 | | |

The number of components involved, for each pump design, ranges between 8 and 12 and the total number of components is 23. Each of the 23 components is assigned a number as shown in Table 3.6. The six BOM trees for the six pumps are reconstructed in phylogenetic-based format as shown in Figure 3.10 and the corresponding encoded matrices for the six BOM trees are shown in Figure 3.11.

The dissimilarity matrix obtained based on the un-normalized dissimilarity measure RF-BOM, using the proposed Integer Programming model, is shown in Table 3.7. It took less than 1 second of computational time to obtain all the values in Table 3.7 on a PC of Intel Core2 Quad 2.83 GHZ processor and 4 GB Ram. The maximum dissimilarity values (RF-BOM$_{max}$) are calculated and given by the matrix shown in Table 3.8. Hence, by substitution in Equation 3.6, the final similarity matrix, based on the normalized similarity measure RF-BOM$_{norm}$, for the six pumps is shown in Table 3.9.

**Pump1**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.33 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Pump2**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0 | 0 | 0 | 0.33 | 0.33 | 0 | 0.5 | 0.5 | 0.5 | 0.33 | 0.33 | 0.33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Pump3**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0.33 | 0 | 0.33 | 0.33 | 0.33 | 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0.33 | 0.33 | 0.33 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0.5 | 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Pump4**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0 | 0 | 0.5 | 0 | 0.5 | 0 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 1 | 0.5 | 0.5 | 0.5 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**Pump5**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 0.5 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Pump6**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0 | 0 | 0.5 | 0 | 0.5 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 1 | 0.5 | 0.5 | 0.5 | 0.5 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**Figure 3.11**: Encoded matrices for the six pumps

**Table 3.7**: BOM trees dissimilarity matrix based on RF-BOM

| **Pump #** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| **1** | - | 14.1 | 10.5 | 15.6 | 8.4 | 18.6 |
| **2** | | - | 14.4 | 17.4 | 17.4 | 20.4 |
| **3** | | | - | 18.9 | 14.8 | 21.9 |
| **4** | | | | - | 15 | 3 |
| **5** | | | | | - | 12 |
| **6** | | | | | | - |

**Table 3.8**: Maximum similarity matrix based on RF-BOM$_{max}$

| **Pump #** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| **1** | - | 33.63 | 37.29 | 32.98 | 28.98 | 32.98 |
| **2** | | - | 36.96 | 32.65 | 28.65 | 32.65 |
| **3** | | | - | 36.31 | 32.31 | 36.31 |
| **4** | | | | - | 28 | 32 |
| **5** | | | | | - | 28 |
| **6** | | | | | | - |

**Table 3.9**: Final BOM trees normalized similarity matrix based on RF-BOM$_{norm}$

| **Pump #** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| **1** | - | 0.580 | 0.718 | 0.527 | 0.710 | 0.436 |
| **2** | | - | 0.610 | 0.467 | 0.393 | 0.375 |
| **3** | | | - | 0.479 | 0.542 | 0.397 |
| **4** | | | | - | 0.464 | 0.906 |
| **5** | | | | | - | 0.571 |
| **6** | | | | | | - |

The corresponding hierarchical clustering tree constructed using Linkage function of MATLAB® is shown in Figure 3.12. It should be mentioned that the obtained grouping results would not change if different order of the pumps is used. According to that tree, if a 70% similarity level, for instance, is desired then the following four pump families are formed: {4, 6}, {1, 3}, {5} and {2}. While if 60% similarity level is required, then the following three pump families are formed: {4, 6}, {1, 3, 5} and {2}. The obtained results are matching the results obtained before in Chapter 2 for the same case study using tree reconciliation, in which the best pump variant that matches pump 1 was sought. Results obtained by both the tree reconciliation based method and Robinson-Foulds based method, without normalizing, agree that pump 5 is the best matching variant for pump 1. Normalization is not needed when it is only required to find the best matching BOM tree to a given one.

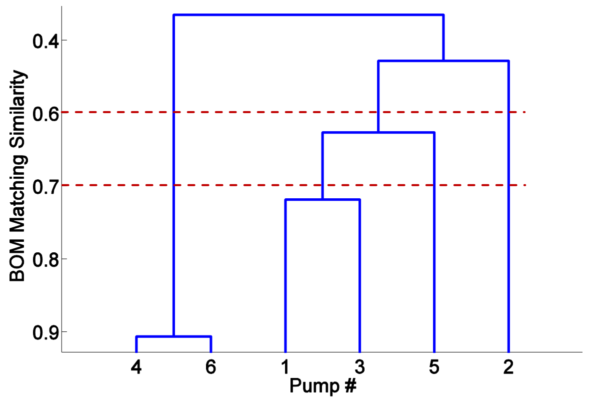


**Figure 3.12**: Dendrogram for the six pumps based on RF-BOM$_{norm}$

## 3.6 Comparison with Jaccard's Commonality Measure

The obtained products grouping/family formation results are compared with those obtained by grouping based on Jaccard's component commonality measure, which is a frequently used similarity measures in product family formation literature. Jaccard's coefficient $J_{ij}$ between any pair of products $i$ and $j$ is given by Equation 3.8, where $a$ is the number of shared components between $i$ and $j$, $b$ is the number of components that are in $i$ but not $j$, and $c$ is the number of components that are in $j$ but not $i$. It is simply the ratio between the number of shared components to the total number of components, regardless of the relationship between components. Table 3.10 shows the obtained similarity matrix for the six pumps using Jaccard's commonality coefficient.

$$J_{ij} = \frac{a}{a+b+c} \qquad (3.8)$$

The dendrogram constructed based on Jaccard's similarity for the six pumps is shown in Figure 3.13. The structural differences between the six BOM trees are not significant; however, they are sufficient for differentiating the resulting clustering tree based on Jaccard's plain component commonality (Figure 3.13) from the clustering tree based on comparing BOMs (Figure 3.12). This is particularly clear in the cluster formed for the three pumps 1, 3 and 5 in each case. Both BOM trees for pumps 1 and 3 have the same number of hierarchical levels and the same number of sub-assemblies. They are more structurally similar than pumps 1 and 5. Although the plain component commonality between pumps 1 and 5 is a little higher than component commonality between pumps 1 and 3; the combined effect represented by RF-BOM$_{norm}$ has favoured the clustering of pumps 1 and 3 then 5 ({1, 3}, {5}) over the clustering of pumps 1 and 5 then 3 ({1, 5}, {3}). The similarity in BOM trees is a reflection of similarity in product assembly structures, thus, from an assembly process point of view; grouping based on RF-BOM$_{norm}$ is more efficient than grouping based on component commonality alone.

**Table 3.10**: Similarity based on components commonality using Jaccard's similarity coefficient

| **Pump #** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| **1** | - | 0.429 | 0.571 | 0.400 | 0.636 | 0.313 |
| **2** | | - | 0.467 | 0.313 | 0.286 | 0.235 |
| **3** | | | - | 0.353 | 0.429 | 0.278 |
| **4** | | | | - | 0.357 | 0.833 |
| **5** | | | | | - | 0.461 |
| **6** | | | | | | - |

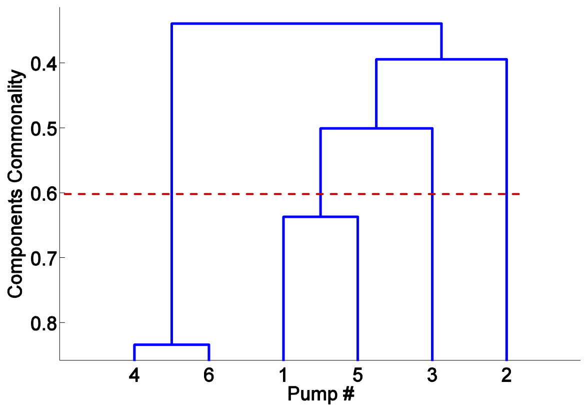


**Figure 3.13**: Dendrogram for the six pumps based on Jaccard's component commonality

Furthermore, RF-BOM$_{norm}$ resulted in higher similarity values than Jaccard's for all pairs of matched pumps. Hence, for the same similarity level, grouping based on RF-BOM$_{norm}$ is more likely to result in less number of families compared to grouping based on Jaccard's. Table 3.11 summarizes the pump families formed based on each grouping criterion at 60% similarity level. Grouping based on RF-BOM$_{norm}$ results in three families, with pump 3 being in the same family with pumps 1 and 5, while based on Jaccard's coefficient, it is recommended to have four families, with pump 3 considered a separate family by itself.

**Table 3.11**: Obtained pump families at 60% similarity based on RF-BOM$_{norm}$ and Jaccard's

| **Commonality measure** | **Family 1** | **Family 2** | **Family 3** | **Family 4** |
|---|---|---|---|---|
| RF-BOM$_{norm}$ | {4, 6} | {1, 3, 5} | {2} | - |
| Jaccard's | {4, 6} | {1, 5} | {2} | {3} |

## 3.7 Summary

In this chapter, the product family formation problem was addressed, in which similarity between BOM trees was used as a grouping criterion. The novelty of the proposed product family formation method is that it uses a similarity measure that considers similarity in the product assembly structure. A new BOM matching measure (RF-BOM$_{norm}$) was developed to overcome some limitations of the tree reconciliation-based method utilized in Chapter 2. The proposed measure is an enhanced version of the Robinson-Foulds (RF) distance employed in Chapter 4 to build the consensus/master assembly tree.

An Integer Programming (IP) model was formulated to mathematically describe the RF-BOM$_{norm}$ measure as well as to calculate it for any given pair of BOM trees. The model is of polynomial time complexity; hence, dealing with instances involving tens of products variants and hundreds of components would feasible. The proposed product family formation method was implemented on the pumps case study used Chapter 2. Accordingly, the pumps were grouped into families in which the obtained results were different from what could be obtained by only considering plain components commonality as a grouping criterion. From an assembly point of view, grouping based on RF-$BOM_{norm}$ resulted in more suitable family formation method. Moreover, grouping based on RF-$BOM_{norm}$ resulted in less number of pump families for the same similarity level than grouping based on Jaccard's component commonality alone.

The developed product family formation method is systematic and the proposed Integer Programming model is fairly simple. They are easy to implement and automate. The required input data is the BOM tree of all involved product variants. However, different engineers may build different BOM tree for the same product within the same company. Thus, the consistency of used BOM trees should be carefully checked before applying the proposed method. Product family formation is often carried out when complete and accurate production cost information, such as machines or assembly stations re-tooling and other setup costs, are readily available. In this study, products related information only is incorporated in the developed method.

# CHAPTER 4
ASSEMBLY SEQUENCE PLANNING BASED ON CONSENSUS TREES

## 4.1 Introduction

The third product development application addressed in this research is assembly sequence planning. The assembly sequence of products has a direct impact on many assembly aspects such as the assembly cost, assembly difficulty, need for fixtures, the likelihood of components damage during assembly and rework, and the ability to do in-process testing (De Fazio and Whitney 1987). Extensive research has been carried out on assembly sequence generation; however, most algorithms do not make the best use of available legacy assembly sequence data. In this chapter, a novel knowledge-based assembly sequencing method that uses a non-traditional approach is introduced. It seeks to determine the master assembly sequence of a family of products to facilitate subsequent product variants assembly sequence planning.

A master assembly sequence is a generic assembly sequence for a group of products which share a number of components and common product structure. Upon building the master assembly sequence for a given family of products, a semi-final assembly sequence for a new variant that falls within, or considerably overlaps with, the scope of the considered family of products could be directly constructed. Developing a master assembly sequence could be considered as a "clustering" type of application for knowledge discovery. The proposed approach is inspired by the problem studied in phylogenetics of building the consensus classification or evolutionary tree for a set of conflicting trees (Adams 1972; Dong et al. 2010). The concept of master assembly sequence has been addressed using different names (Gupta and Krishnan 1998; Martinez et al. 2000; Lai and Huang 2003). For example, Martinez et al. (2000) generated a master assembly sequence, called parent plan, for a hypothetical product which includes all components of a product family (called meta-product). However, they employed a traditional assembly sequence planning approach (reviewed in Section 4.2) to generate the master assembly sequence for the meta-product.

In this research, a binary rooted tree is used to represent any assembly sequence including the master assembly sequence. A new MIP (Mixed-Integer Programming) model, as well as a custom-designed Genetic Algorithm, has been developed to find the optimal consensus tree. The proposed model is demonstrated using a case study of control valves where the master (consensus) assembly sequence for existing valve variants is derived and used to form the assembly sequence of a new variant.

## 4.2 Literature Review

Significant research has been carried out on assembly sequence generation (ElMaraghy and Knoll 1991; ElMaraghy and Laperriere 1992; ElMaraghy and Rondeau 1992; Laperriere and ElMaraghy 1994; Dini et al. 1999; Sinanoglu and Borklu 2005; Azab et al. 2008; Huang et al. 2008; Wang and Liu 2010; Jimenez 2013). Research on automating or semi-automating assembly sequence generation goes back to the eighties and earlier (Whitney 2004). However, the main solution principles adopted by most algorithms are more or less the same.  A conventional assembly sequence generation algorithm has two phases. First, the set of all feasible assembly sequences is generated according to a pre-defined set of feasibility/validity constraints. Geometrical and precedence feasibility of an assembly operation represent the major feasibility constraints. Second, the set of feasible sequences generated is searched for the best sequence according to specified optimization criteria such as minimum total number of changes of parts orientation and assembly tools during assembly (Whitney 2004).

Instead of generating the assembly sequence in two consecutive steps, the Generative Assembly Process Planner (GAPP) developed by Laperriere and EIMaraghy (1994) used an A* algorithm (Nilsson 1980) which simultaneously applies both feasibility constraints (geometric and accessibility constraints) and optimization criteria (number of parts re-orientations, concurrent execution of assembly tasks, grouping of similar tasks and assembly stability). GAPP finds the optimal assembly sequence without exhaustively generating all feasible sequences.

Many recent algorithms merge the two phases by utilizing a fitness function that considers both assembly constraints and optimization criteria using a meta-heuristic optimization algorithm. A typical example is the assembly sequence generation algorithm, based on Particle Swarm Optimization (PSO), by Wang and Liu (2010) which studied  the linear assembly sequencing problem where parallel assembly operations are not allowed (i.e. all product components are assembled one at a time) (Whitney 2004). A linear assembly sequence is normally encoded by a permutation (sequence) vector the length of which equals the number of assembly components. The product is then analyzed and five component-to-component relational matrices are constructed. The first three matrices represent geometric, local assembly precedence and assembly stability feasibility constraints that should be satisfied in any final assembly sequence. The other two matrices represent two assembly optimization criteria that may favor one sequence over another. These are the number of assembly tool changes and number of assembly connection changes in a generated assembly sequence. The number of assembly direction changes is also integrated as a third optimization criterion through the same matrix used to represent the geometric constraint. A discrete Particle Swarm Optimization (PSO) algorithm is then used to search for the best assembly sequence using a weighted fitness function of the incorporated constraints and optimization criteria. The algorithm is schematically described in Figure 4.1.

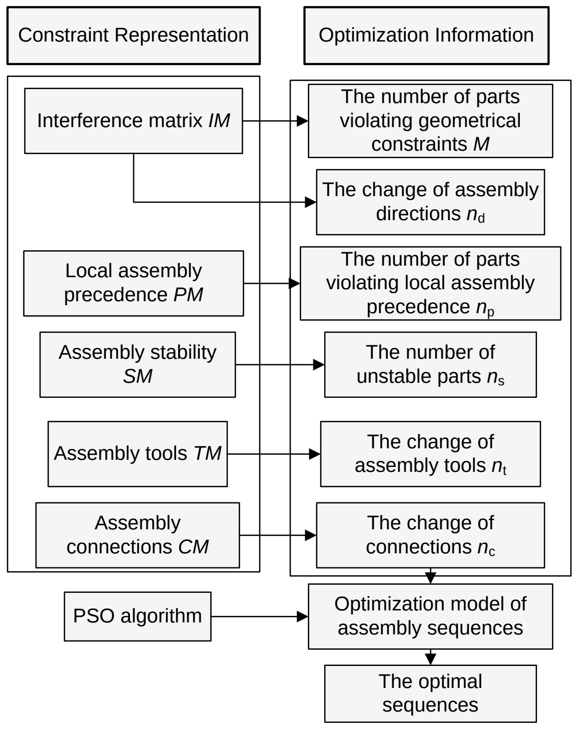

Constraint Representation
Optimization Information
Interference matrix *IM*
The number of parts violating geometrical constraints *M*
The change of assembly directions $n_d$
Local assembly precedence *PM*
The number of parts violating local assembly precedence $n_p$
Assembly stability *SM*
The number of unstable parts $n_s$
Assembly tools *TM*
The change of assembly tools $n_t$
Assembly connections *CM*
The change of connections $n_c$
PSO algorithm
Optimization model of assembly sequences
The optimal sequences

**Figure 4.1**: A typical assembly sequence generation algorithm (reproduced from (Wang and Liu 2010))

The work of Laperriere and EIMaraghy and Wang and Liu are only few instances of how assembly sequencing has been normally addressed for decades. Several variants of these algorithms are found in literature. Among the key elements that distinguish an algorithm is for instance the assembly sequence representation form. Other common forms of assembly sequence representation include AND/OR graphs (Homem de Mello and Sanderson 1989), assembly states (Laperriere and ElMaraghy 1994), and partial assembly trees (Miller and Hoffman 1989). Many other optimization techniques/algorithms have been utilized such as Artificial Neural Networks (ANN) (Sinanoglu and Borklu 2005), Genetic Algorithm (GA) (Marian et al. 2002), Simulated Annealing (SA) (Hong and Cho 1997), and Ant Colony Optimization (ACO) (Wang et al. 2005). Some recent algorithms used CAD models in automatically generating and applying assembly constraints employing sophisticated geometry handling algorithms (Pan et al. 2006). Assembly constraints may also be pre-defined by the planner by pre-analyzing the product (Wang and Liu 2010) or using interactive questions-and-answers (De Fazio and Whitney 1987). Semi-automatically generated assembly constraints would be practical with simpler and less complicated products. The number and type of assembly constraints and optimization criteria incorporated by an assembly sequence generation algorithm vary depending on assembly facility, product type, assembly volume, and other conditions. A thorough review for assembly constraints and optimization criteria  is found in (Jones et al. 1998).

Existing methods for retrieval or knowledge-based assembly sequence generation rely on retrieving the most similar existing product variant with respect to commonality of product components and assembly structure (Dong et al. 2005). The assembly sequence plan of such a retrieved product variant is then used as a preliminary one for the new variant. Matching products Bill of Material (BOM) trees is one way of achieving this goal (Shih 2011). The assembly sequence data (i.e. sequencing relationships between components) for a new combination of components might be completely available in the manufacturer's database but in a scattered form, where overlapping sub-groups of that new combination of components are distributed among different product variants. Thus, the critical limitation of individual assembly sequence retrieval methods is that the assembly sequence data for a given new combination of components that has never existed together before in an existing variant could not be captured. This is what the proposed approach, based on the concept of master assembly sequence tree, is designed to overcome.

## 4.3 Research Scope and Methodology

The proposed approach looks for the consensus (master) assembly sequence that best represents the set of available individual sequences even if some conflicts among them exists; for example different assembly sequence may exit for the same combination of components. The consensus assembly sequence is seeking to capture and aggregate together all common (e.g. most frequent) assembly sequencing relationships available. Such a consensus (master) assembly sequence would then be used to construct assembly sequences for new product variants. The proposed assembly generation method is suitable for mechanical assembled assemblies such as engines, pumps, valves, etc. Given a set of *N* assembly sequences, for *N* product variants with a total of *n* different components, it is required to find a single assembly sequence of all the *n* components with the minimum total dissimilarity with existing *N* assembly sequences. Measuring dissimilarity between individual sequences will be detailed in Section 4.4.1. The following assumptions are made in this chapter:

- Assembly sequence data for existing product variants are available.
- Non-linear assembly with parallel operations is allowed.
- Assembly operations are sequential with one component added at a time.
- The same name or part number is used for various versions or variants of the same component.

Non-linear assembly is the general case of linear assembly and therefore linear assembly also falls within the scope of the proposed model. The assembly sequence of a non-linear assembly can be represented by un-ordered rooted binary trees or what is known in the assembly sequencing literature as partial assembly trees (Wolter 1992). The root of a partial assembly tree represents the final product (complete assembly) and the leaves represent individual components. Other intermediate nodes represent the sub-assembly resulting from adding its branches which may be sub-assemblies, components, or a combination of sub-assemblies and components. Partial assembly trees are different from BOM trees used in Chapters 2 and 3. BOM trees are just a representation form for the product contents and structure that does not provide information about assembly precedence constraints like partial assembly trees. In reality, partial assembly trees could be constructed from assembly process sheets which are used in assembly plants to describe and document the entire assembly procedure. Those sheets typically include data about the number, description and sequence of assembly operations as well as the required equipment and tooling. Assembly process sheets are typically available in the form of electronic spreadsheets created, stored and managed by the assembly planning department.

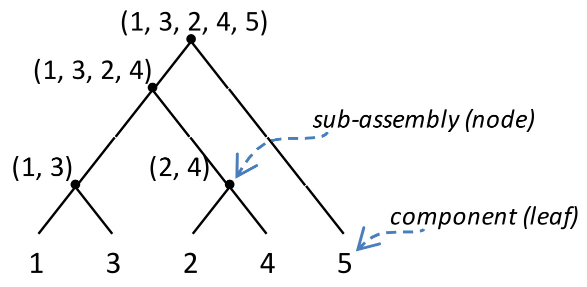


**Figure 4.2**: A partial assembly tree for five components

Figure 4.2 shows the partial assembly tree representing the assembly sequence of a product consisting of five components. According to this plan, components 1 and 3 may be assembled before, after, or at the same time as components 2 and 4. Component 5 is then added to sub-assembly (1, 3, 2, 4) to obtain the complete product (1, 3, 2, 4, 5). It is then required to find the *consensus* tree that best represents a given set of partial assembly trees or *cladograms*. The two terms “consensus” and “cladograms” are adopted from the biology and phylogenetics literature where these tools are used. A cladogram, as introduced before in chapter 2, is a phylogenetic tree which basically has the same structure and properties as a partial assembly tree and is used in biological studies to classify species and analyze their evolutionary histories (Page 1994). It is not unusual in some biological science and evolutionary studies fields to have different classifications for the same set of species, where a consensus tree (Adams 1972) for these classifications is sought.

## 4.4 Knowledge-Based MIP Model for Constructing the Master Assembly Tree

Many algorithms and methods have been developed in the biology and phylogenetics literature to construct consensus trees; the term "consensus tree" itself has different definitions depending on the applied consensus method such as strict consensus and majority rule consensus (Bryant 2003). The consensus tree is defined in this chapter as the tree that has the minimum sum of dissimilarity distances from each of the individual trees. Such a distance is determined using a commonly used measure in phylogenetics known as *Robinson-Foulds* distance (Robinson and Foulds 1981). In addition, the consensus tree must be a binary tree which includes all components of the individual trees and each component appears only once.

Based on this definition, a mathematical Integer Programming model was developed, for constructing the consensus tree of a set of individual partial assembly trees (assembly sequence trees or assembly trees for short) that do not necessarily share the identical set of components. Most existing methods for constructing the consensus tree deal with trees with the exact same set of leaves, which is rarely the case for assembly sequence trees. The developed model was initially developed as a non-linear model then it was linearized. Before introducing both the non-linear and linearized formulations, the next two sections first describe how the dissimilarity distance between any given two trees is quantified as well as how a given assembly sequence tree is encoded using the developed model.

### 4.4.1 Robinson-Foulds Distance

The Robinson-Foulds (RF) distance (Robinson and Foulds 1981) is the most widely used metric for comparing phylogenetic trees (Pattengale et al. 2007). Given two trees *T1* and *T2*, both having *m* number of leaves, *S1* is a set that includes *m-1* subsets each representing one of the *m-1* nodes of *T1* and the elements inside each subset are the elements belonging to the node representing the subset. Similarly, *S2* contains *m-1* subsets representing the *m-1* nodes of *T2*. Robinson-Foulds distance (RF) is given by Equation 4.1, where "Δ" refers to symmetric difference (a set theory operation). Equation 4.1 could be further detailed as in Equation 4.2, where "\" refers to set difference operation. Hence, RF is simply a normalized count of the nodes (i.e. clusters of leaves) that exist in one tree, but not the other. Robinson-Foulds is an reliable measure for comparing phylogenetic trees, in which it satisfies the following four conditions (Cardona et al. 2009) that formally qualifies it as a metric in a mathematical sense:

1. Non-negativity: $RF\ (T1, T2) \geq 0$
2. Separation: $RF(T1, T2) = 0$, if and only if $T1 \cong T2$

3. Symmetry: *RF(T1, T2) = RF(T2, T1)*
4. Triangle inequality: *RF(T1, T3)* ≤ *RF (T1, T2)* + *RF (T2, T3)*

$$RF\ (T1, T2) = \tfrac{1}{2}\ |S1\ \Delta\ S2| \qquad (4.1)$$

$$RF\ (T1, T2) = \tfrac{1}{2}\ (|S1 \setminus S2| + |S2 \setminus S1|) \qquad (4.2)$$

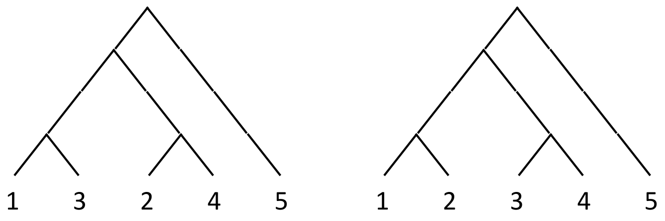


**Figure 4.3**: Two trees *T1* and *T2* with *RF (T1, T2)* = 2

For instance, the two trees *T1* and *T2* shown in Figure 4.3 each has five leaves and four nodes. For *T1*, *S1* = {{1, 3, 2, 4, 5}, {1, 3, 2, 4}, {1, 3}, {2, 4}} and for *T2*, *S2* = {{1, 2, 3, 4, 5}, {1, 2, 3, 4}, {1, 2}, {3, 4}}. The order of sets within *S1* and *S2* or order of elements within any of their subsets is not important. By substituting in Equation 4.2, we get *RF (T1, T2)* = ½ (2 + 2) = 2.

### 4.4.2 Encoding Assembly Sequence Trees

A matrix form is used to encode assembly trees for the developed model. In encoding a tree of *m* leaves into a matrix, the information to be represented by the matrix is the grouping and hierarchical relationships of the *m* leaves inside the tree. This is equivalent to the leaves which should belong to the same node. This is carried out through the proposed tree-to-matrix encoding scheme shown in Figure 4.4. A binary rooted tree of *m* components has *m-1* nodes; thus, for a tree of *m-1* nodes a matrix of *m-1* rows and *m* columns is used to represent it (i.e. a matrix of size (*m-1)* x *m*). Columns of the matrix stand for components and rows stand for nodes. Binary (0-1) values are used to represent the presence or absence of a given component in any given node.

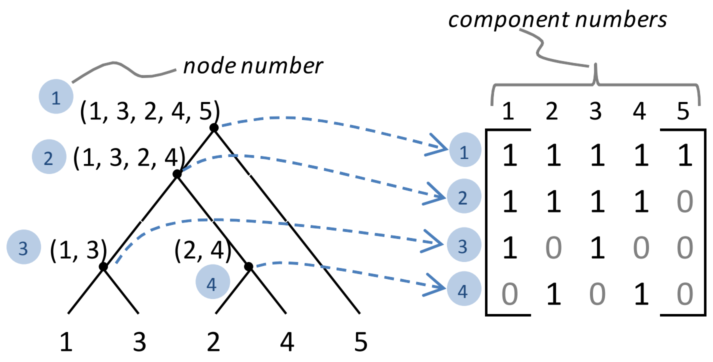


**Figure 4.4**: Proposed tree-to-matrix encoding scheme

For instance, node number 2 in the tree in Figure 4.4 has the components 1, 2, 3, and 4, thus, row number 2 in the corresponding encoded matrix will take the value 1 in the cells number 1, 2, 3, and 4, while cell number 5 will have a 0 value due to the absence of that component (component 5) in that node (node 2). Therefore, any encoded matrix will always have one of its rows full of ones so as to represent the root node to which all components belong. Throughout the proposed model, the first row of any matrix is always reserved for the root node and hence all its cells are ones. This is how the mathematical model can easily identify all components involved in a given tree.

### 4.4.3 Non-Linear Formulation

This section introduces the formulation of the proposed model for generating the master (consensus) assembly sequence tree for a given set of individual assembly sequence trees. The formulation is for a non-linear Integer Programming model with non-linear objective function and one quadratic constraint. In the next section the non-linear parts of the model are linearized and a reformulated Mixed-Integer Linear Programming model will be presented. The model input parameters are:

| | |
|---|---|
| $N$ | Number of individual trees |
| $n$ | Total number of different components |
| $d$ | Number of nodes of the master tree ($n$-1) |
| $m_i$ | Number of nodes in tree $i$ |
| $t_{ijk}$ | A binary (0-1) element in the $j$th row and $k$th column of the encoded matrix representing the $i$th tree (e.g. the matrix in Figure 4.4) |

And the decision variables are:

| | |
|---|---|
| $cm_{uj}$ | A binary (0-1) element in the $u$th row and $j$th column of the sought consensus/master matrix |
| $x_{uv}$ | A binary (0-1) element valued at 1 if node $v$ is directly branched from node $u$ in the master matrix |
| $y_{uj}$ | A binary (0-1) element valued at 1 if component $j$ is directly branched from node $u$ in the master matrix |

Using the listed parameters and decision variables, a non-linear Integer Programming model was formulated with four sets of constraints as follows:

$$Minimize \sum_{i=1}^{N}\sum_{k=1}^{m_i} \min_{u=1\to d} \left(\sum_{j=1}^{n}\left|(t_{ijk} - cm_{uj})\right| t_{i1j}\right) \quad (4.3)$$

$$Subject\ to$$

$$cm_{uj} = y_{uj} + \sum_{v=1|v\neq u}^{d} cm_{vj}x_{uv} \qquad \forall\, u = 1,\dots,d \quad \forall\, j = 1,\dots,n \qquad (4.4)$$

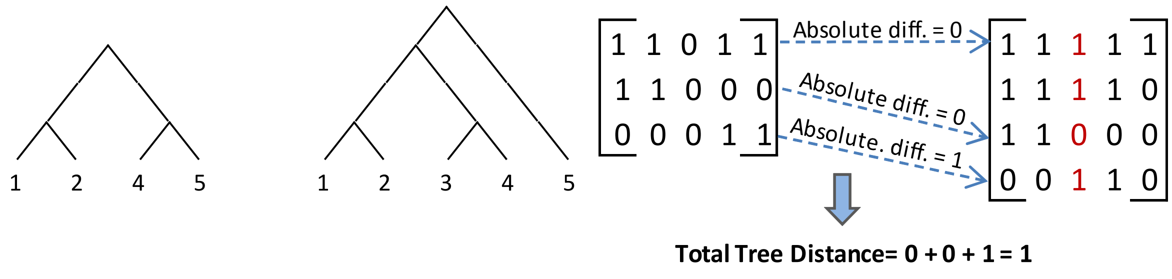
$$\sum_{v=1|v\neq u}^{d} x_{uv} + \sum_{j=1}^{n} y_{uj} = 2 \qquad \forall\, u = 1,\dots,d \qquad (4.5)$$

$$\sum_{u=1}^{d} y_{uj} = 1 \qquad \forall\, j = 1,\dots,n \qquad (4.6)$$

$$\sum_{u=1|v\neq u}^{d} x_{uv} = 1 \qquad \forall\, v = 2,\dots,d \qquad (4.7)$$

The objective function (Equation 4.3) minimizes the total Robinson-Foulds (RF) distance between each individual tree and the sought master tree. It calculates the RF distance between each individual tree and the sought master tree and then sum up all these distances. As shown in Equation 4.3, the objective function is a non-linear expression that accomplishes three tasks. First, the objective function maps each node in a given individual tree to the most similar node in the master tree. This is realized by using the inner minimization function (*min*). Second, it quantifies the difference between each row (node) in any given individual tree and its corresponding mapped row in the master tree. This is realized through the cell-by-cell subtraction operations, as well as the use of the absolute function so as to count every difference whether it's positive or negative. Third, in measuring the distance between a given individual tree of *q* components and a master tree of *n* components it excludes all (*n* - *q*) components that exist in the consensus tree but not the individual trees. This is accomplished by multiplying the subtraction term by $t_{i1j}$. This is the first row, in a given individual tree *i*, which represents the set of components existing in that tree.

To further explain the objective function, consider the two trees shown in Figure 4.5 (a). The first tree is one of the existing individual trees (an input) and the second tree is a potential master tree. Figure 4.5 (b) shows the corresponding matrix representation of both trees and the calculated distance according to the formulated objective function (Equation 4.3). Third column elements (shown in different color) will not be considered because the third component (component # 3) does not exist in the individual tree. Doted arrows refer to the rows (nodes) from the master matrix that best match rows from the individual tree. The tree distance measured by the proposed objective is considered as an "RF-like" measure; it applies the same concept and leads to the same decision but does not always give the same values. The reason for modifying the original RF metric was to simplify the objective function. The modified RF measure has been also shown, using various test examples, to be a metric that satisfies the four metric axioms stated earlier in Section 4.4.1.



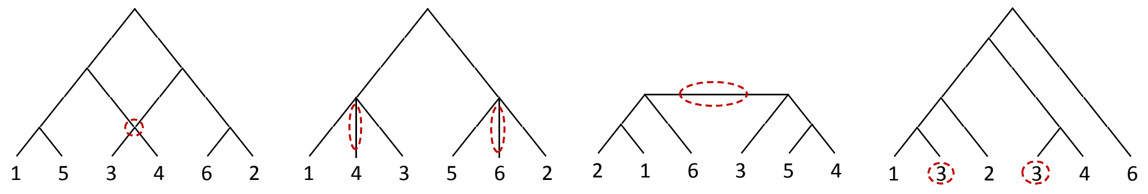
(a) (b)

**Figure 4.5**: An example illustrating the objective function

Four sets of constraints, in addition to the binary variables constraints, are used to ensure the generation of valid master trees (feasible solutions). Accordingly, the first set (Equation 4.4) states that any node, represented by a row in the matrix, should be the result of combining other nodes in the matrix (second term in the LHS) and/or components (first term in the LHS). The second set (Equation 4.5) ensures that each node is the combination of exactly two branches; either two other nodes, two components or one other node and one component. The third set of constraints (Equation 4.6) states that each component must branch out directly from one single node; no component could be branched from two or more different nodes at the same time. Similarly, the fourth set of constraints (Equation 4.7) ensures that every node, except the root node (node # 1), is directly branched from another node; no node could be branched from two or more different nodes at the same time. The combination of these four sets of constraints preserves the following characteristics in any generated master tree:

- Hierarchical tree structure,
- binary,
- rooted, and
- Unique existence of all components.

Figure 4.6 exhibits some examples of invalid master trees, when some or all constraints are not satisfied for a data set that involves 6 components (from 1 to 6).



A tree structure is not realized Binary feature missed Rooted feature missed Repeated and missed components

**Figure 4.6**: Examples of invalid trees

### 4.4.4 Linearized Mixed-Integer Formulation

Three operations (Bisschop 2005) are carried out to linearize the presented non-linear formulation; two of them are performed on the objective function to linearize the *absolute* and *min* functions and the third is to linearize the first quadratic constraint. The absolute function for a binary variable is linearized by first changing the absolute function into a square function which has the same output for a binary variable. Then, the linearized form of the square of any binary variable $x^2$ is simply $x$. Thus the linearized form of the expression $\left|(t_{ijk} - cm_{uj})\right|$ (absolute difference between two cells) in the objective function becomes:$(t_{ijk} - 2cm_{uj}t_{ijk} + cm_{uj})$.

In order to linearize the *min* function performed over the node-difference expression $\sum_{j=1}^{n}\left|(t_{ijk} - cmuj)ti1j\right.$ for all $u$ = 1 to number of nodes ($d$), the following two new variables are defined:

| | |
|---|---|
| $Z_{iku}$ | A binary (0-1) variable that takes 1 if node $k$ in the individual tree $i$ is to be mapped to the node $u$ from the master tree. |
| $w_{iku}$ | An auxiliary positive variable that takes the value of the expression $\sum_{j=1}^{n}\left|(t_{ijk} - cm_{uj})\right|t_{i1j}$ (node-difference expression) when only $Z_{iku}$ is equal to 1. |

Then, a new objective function (Equation 4.8) along with two constraints (Equations 4.9 and 4.10) are defined as follow, where $M$ in Equation 4.9 is a large number (a rough safe estimate for $M$ is to be bigger than $n$):

$$Minimize \sum_{i=1}^{N}\sum_{k=1}^{m_i}\sum_{u=1}^{d} w_{iku} \tag{4.8}$$

$$w_{iku} + M(1 - Z_{iku}) \geq \sum_{j=1}^{n}(t_{ijk} - 2cm_{uj}\, t_{ijk} + cm_{uj})\, t_{i1j} \qquad \begin{array}{l}\forall\, i = 1, \dots, N \\ \forall\, k = 1, \dots, m_i \\ \forall\, u = 1, \dots, d\end{array} \tag{4.9}$$

$$\sum_{u=1}^{d} Z_{iku} = 1 \qquad \begin{array}{l}\forall\, i = 1, \dots, N \\ \forall\, k = 1, \dots, m_i\end{array} \tag{4.10}$$

The new objective function minimizes total sum of all the $w$ variables (node difference variables) and the two new sets of constraints ensure that every node $k$ in every individual tree $i$, is only mapped to one single node $u$ in the master tree. Therefore, each node in any given individual tree will always be mapped to its best matching node in the master tree so as to achieve the minimum value of the new objective function (Equation 4.8). The final linearization operation is performed on the first set of quadratic constraints (Equation 4.4). To linearize the multiplication of the two binary variables $cm_{vj}$ and $x_{uv}$, the following new binary variable is defined:

$r_{ujv}$ A binary (0-1) variable that takes 1 if only the two binary variables $cm_{vj}$and $x_{uv}$ are both equal to 1.

Then, a new four sets of constraints are defined as follows:

$$cm_{uj} = y_{uj} + \sum_{v=1|v\neq u}^{d} r_{ujv} \quad \forall\, u = 1,\dots,d \quad \forall\, j = 1,\dots,n \qquad (4.11)$$

$$r_{ujv} \le cm_{vj} \quad \forall\, u = 1,\dots,d \quad \forall\, j = 1,\dots,n \quad \forall\, v = 1,\dots,d|v \neq u \qquad (4.12)$$

$$r_{ujv} \le x_{uv} \quad \forall\, u = 1,\dots,d \quad \forall\, j = 1,\dots,n \quad \forall\, v = 1,\dots,d|v \neq u \qquad (4.13)$$

$$r_{ujv} \ge cm_{vj} + x_{uv} - 1 \quad \forall\, u = 1,\dots,d \quad \forall\, j = 1,\dots,n \quad \forall\, v = 1,\dots,d|v \neq u \qquad (4.14)$$

The first set of constraints (Equation 4.11) maintains the same role of original constraint (Equation 4.4); while the other three sets of constraints (4.12, 4.13 and 4.14) are used to link the values of the new *r* variable to the multiplication product of the two original *cm* and *x* variables, in which an *r* variable is allowed to have a value of 1 when only both *cm* and *x* are equal to 1. One further set of constraints is added which is not mandatory, but is believed to improve the performance of the model. The additional set of constraints (Equation 4.15) states that the first row in any generated master matrix should be full of ones; this is the root node that contains all the components. This is just to save time of defining which of the nodes to be the root node, which in effect reduces the number of variables in a mathematical sense.

$$cm_{1j} = 1 \quad \forall\, j = 1,\dots n \qquad (4.15)$$

One final adjustment is made to improve the computational performance. Whenever the two indices *u* and *v* are used in the model, the condition $v \neq u$ is always used, where the values of *u* and *v* are both from 1 to total number of nodes in the master tree (*d*). In Equation 4.2, for example, this condition was applied to ensure that each node *u* in the master matrix is the sum of other components and/or nodes except the node *u* itself. The number of constraints in the model could be greatly reduced if the more tightening condition $v > u$ is used instead of $v \neq u$ whenever applicable. For the same mentioned example, this means that each node *u* in the master matrix should be the sum of other components and/or nodes that appear below node *u* in the matrix. This should increase the model efficiency by reducing the symmetry caused by the fact that many matrix configurations could represent the same tree.

Additional restriction which is equivalent to further reduction in the number of constraints and symmetry could be achieved by placing an upper limit *l* which is problem-dependent parameter, on the difference between *u* and *v* (i.e. $0 < v - u \leq l$). The default value for *l* is the number of nodes of the master tree (*d*). The final linearized model is a Mixed-Integer Programming model with ten sets of constrains. The following is its complete formulation where some re-arrangements were performed following the mathematical programming convention of placing all decision variables on the left hand side of the constraints:

$$\text{Minimize} \sum_{i=1}^{N}\sum_{k=1}^{m_i}\sum_{u=1}^{d} w_{iku} \tag{4.8}$$

$Subject\ to$

$$w_{iku} + M(1 - Z_{iku}) - \sum_{j=1}^{n}(t_{ijk} - 2cm_{uj}\, t_{ijk} + cm_{uj})\, t_{i1j} \geq 0 \qquad \forall\, i = 1, \dots, N \quad \forall\, k = 1, \dots, m_i \quad \forall\, u = 1, \dots, d \tag{4.9}$$

$$\sum_{u=1}^{d} z_{iku} = 1 \qquad \forall\, i = 1, \dots, N \quad \forall\, k = 1, \dots, m_i \tag{4.10}$$

$$cm_{uj} - y_{uj} + \sum_{v=1|0<v-u\leq l}^{d} r_{ujv} = 0 \qquad \forall\, u = 1, \dots, d \quad \forall\, j = 1, \dots, n \tag{4.11}$$

$$r_{ujv} - cm_{vj} \leq 0 \qquad \forall\, u = 1, \dots, d \quad \forall\, j = 1, \dots, n \quad \forall\, v = 1, \dots, d | 0 < v - u \leq l \tag{4.12}$$

$$r_{ujv} - x_{uv} \leq 0 \qquad \forall\, u = 1, \dots, d \quad \forall\, j = 1, \dots, n \quad \forall\, v = 1, \dots, d | 0 < v - u \leq l \tag{4.13}$$

$$r_{ujv} - cm_{vj} - x_{uv} \geq -1 \qquad \forall\, u = 1, \dots, d \quad \forall\, j = 1, \dots, n \quad \forall v = 1, \dots, d | 0 < v - u \leq l \tag{4.14}$$

$$\sum_{v=1|0<v-u\leq l}^{d} x_{uv} + \sum_{j=1}^{n} y_{uj} = 2 \qquad \forall\, u = 1, \dots, d \tag{4.5}$$

$$\sum_{u=1}^{d} y_{uj} = 1 \qquad \forall\, j = 1, \dots, n \tag{4.6}$$

$$\sum_{u=1|0<v-u\leq l}^{d} x_{uv} = 1 \qquad \forall\, v = 2, \dots, d \tag{4.7}$$

$$cm_{1j} = 1 \qquad \forall\, j = 1, \dots n \tag{4.15}$$

## 4.5 Illustrative Example

An example is presented in this section to illustrate the studied problem as well as the proposed knowledge-based MIP model for generating master assembly sequence trees. Consider five assembly sequences for five variants in a given product family ($N = 5$) with a total of eight different components ($n = 8$). It is required to derive the master assembly sequence tree of this product family, and then construct the assembly sequence of a new variant which has a new combination of those components. The trees representing the assembly sequences for this product family are shown in Figure 4.7 and the corresponding encoded matrices ($t$ parameters) are shown in Figure 4.8, where columns represent components and rows represent nodes.

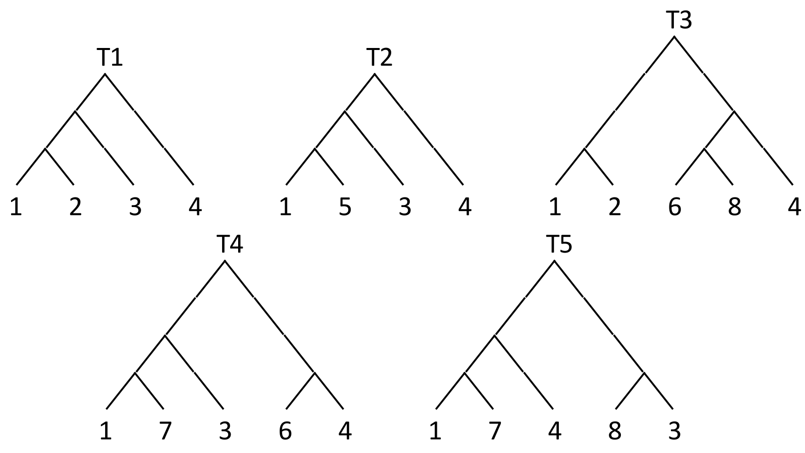


**Figure 4.7**: Assembly sequence trees for family of five variants

T1
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

T2
$$\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

T3
$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

T4
$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

T5
$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

**Figure 4.8**: Encoded matrices for the five assembly sequence trees shown in Figure 4.7

The proposed knowledge-based MIP model is written in AMPL - A Mathematical Programming Language (http://ampl.com/). The optimal master matrix (Figure 4.9) with an optimal (minimum) objective function of 2 is obtained for this example using Gurobi Optimizer 5.5 solver (http://www.gurobi.com/). Obtaining a non-zero objective function means that the existing assembly sequence data contains some inconsistency. It can be seen that the assembly sequence of components 3 and 4 in tree T5 is different from other trees. In order to minimize the objective function value, the model assigns both components to the most common position they used to have in the available data. Hence, when the generated master tree is compared to each individual tree, a zero tree distance is obtained for all trees except T5. Having conflicting data is not uncommon; this is part of the legacy products assembly sequence plans reality that should be dealt with.

Decoding the obtained matrix back into a tree form may be done either manually or by utilizing cladistic analysis software packages that are able to build the most parsimonious or optimal classification tree (cladograms) for a given set of species (Fitch 1971). The software employed in this research is PHYLIP 3.69 (http://www.phylip.com/); a well-known free access software for inferring and analyzing phylogenies.

$$
\text{node numbers}\left\{
\begin{array}{c|cccccccc|}
 & \multicolumn{8}{c}{\overbrace{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}^{\textit{component numbers}}} \\
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
3 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
4 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
5 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{array}\right.
$$

**Figure 4.9**: Master assembly sequence matrix obtained by the MIP model for the above example

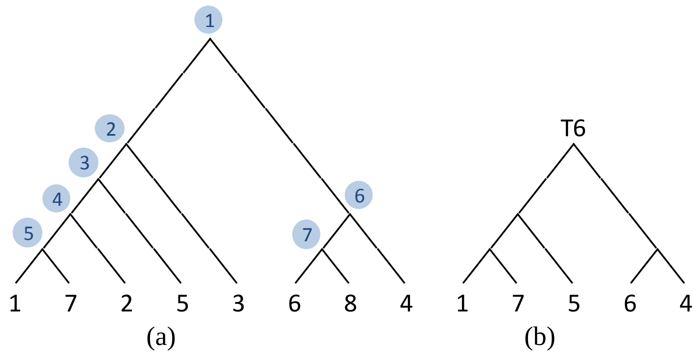


**Figure 4.10**: (a) Master assembly sequence tree and (b) Extracted tree for a new product variant

The *pars* function (comes from parsimony) of PHYLIP 3.69 builds a cladogram for a given set of species featuring different biological characters out of a predefined set of characters. The data format required by this function has exactly the same binary matrix format as the master matrix shown in

Figure 4.9, where tree nodes (i.e. sub-assemblies) stand for species and components stand for biological characters. The obtained tree for the master matrix generated by the MIP model is shown in Figure 4.10 (a) where the tree nodes represent subassemblies and the leaves represent components.

This consensus tree (Figure 4.10 (a)) is the master assembly sequence tree for the considered product family. The assembly sequence tree for a new product variant that contains components 1, 4, 5, 6, and 7 could then be extracted from the tree in Figure 4.10 (a) by including those components in the master tree which exist in the new product. Accordingly, the assembly sequence tree for the new product variant (T6) is shown in Figure 4.10 (b). Assembly sequence trees obtained for new product variants using the proposed model should be revised by the planner, especially when the generated master assembly sequence tree has a non-zero tree distance from any of the individual trees (i.e. much attention is needed when conflicting data exist).

$$\underset{\text{node numbers to which a component directly belongs}}{y \text{ variables}} \quad \overbrace{\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}}^{\text{component numbers}}$$

$$\underset{\text{node numbers to which a node directly belongs}}{x \text{ variables}} \quad \overbrace{\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}}^{\text{node numbers}}$$

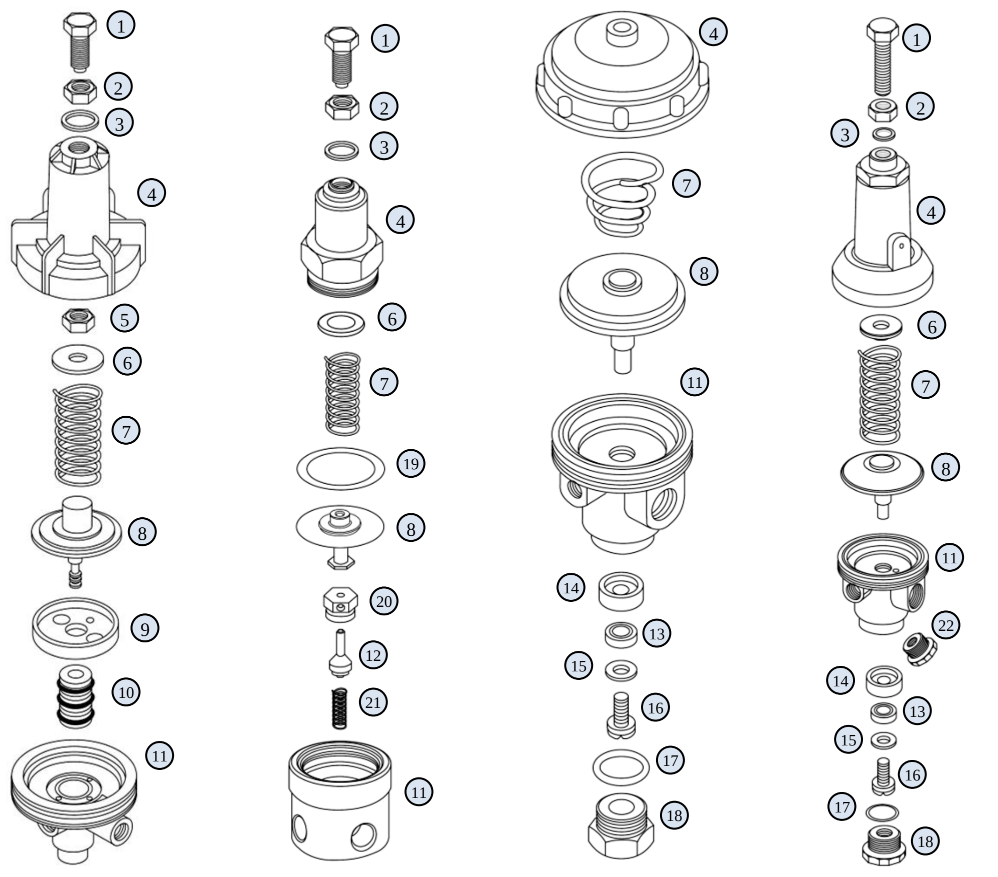
**Figure 4.11**: Resulting *x* and *y* variables values

Figure 4.11 shows the resulting values of both *y* and *x* variables in this example. The *y* matrix simply defines components directly branched from each node in the master matrix; while, the *x* matrix defines the nodes directly branched from each node. This further clarifies the four sets of constraints used to build the original non-linear version of the model. It should be noted that the master matrix shown in Figure 4.10 (a) can be constructed if *x* and *y* matrices are both known; as outlined mathematically by the first set of constraints (Equation 4.4). The other three sets of constraints (Equations 4.5, 4.6, and 4.7) apply validity constraints on the structure of the two matrices as well as the relationships between them.

## 4.6 Case Study

The proposed knowledge-based MIP model is demonstrated using a real case study involving four different variants of a pilot control valve adopted from Dorot® product catalogues (http://www.dorot.com/). A pilot valve is a small valve used to operate larger main control valves. It is typically used in process industries such as chemical, paper and pharmaceutical industries. The company produces several pilot valve variants with various designs for various operating requirements. Many of these valve variants do have a lot of common components and assembly structure.

Three out of the four pilot valve variants involved in this case study are considered as legacy designs by the manufacturer with known and consistently developed assembly sequence plans (variants # 29-200, 68-410 and 66-200). The three valve variants are schematically shown in Figure 4.12 (a), (b), and (c) where the first variant has 11 components, the second variant has 12 components and the third variant has 10 components. The fourth pilot valve (variant # 66-300 in Figure 4.12 (d)), composed of fifteen components, is considered to be the new variant for which an assembly plan is required.



**(a) Variant # 29-200** **(b) Variant # 68-410** **(c) Variant # 66-200** **(d) Variant # 66-300**

**Figure 4.12**: Exploded views for the family of pilot valves extracted from the manufacturer catalogues (http://www.dorot.com/). (a), (b), and (c) are considered as existing ones while (d) is considered as new one

The total number of different components involved in the case study is 22. Components names are listed in Table 4.1. Figure 4.13 shows the assembly sequence trees (T1, T2 and T3) extracted for each three existing valve variant from the exploded views provided in the manufacturer catalogue. The objective is then to construct the master assembly sequence for this family of valves and to further use it to generate the assembly sequence of the new valve variant.

**Table 4.1**: Components names for the studied family of pilot valves

| No. | Component Name | No. | Component Name |
|---|---|---|---|
| 1 | Adjusting Bolt | 12 | Seal Assembly |
| 2 | Locking Nut | 13 | Seal |
| 3 | Spring Ring | 14 | Seal Housing |
| 4 | Bonnet | 15 | Seal Disc |
| 5 | Bonnet Nut | 16 | Seal Bolt |
| 6 | Spring Disc | 17 | Seat Washer |
| 7 | Spring | 18 | Seat |
| 8 | Diaphragm Assembly | 19 | Bonnet Washer |
| 9 | Diaphragm Bowl | 20 | Nozzle |
| 10 | Ports Selection Body | 21 | Balancing Spring |
| 11 | Body | 22 | Outer Plug |

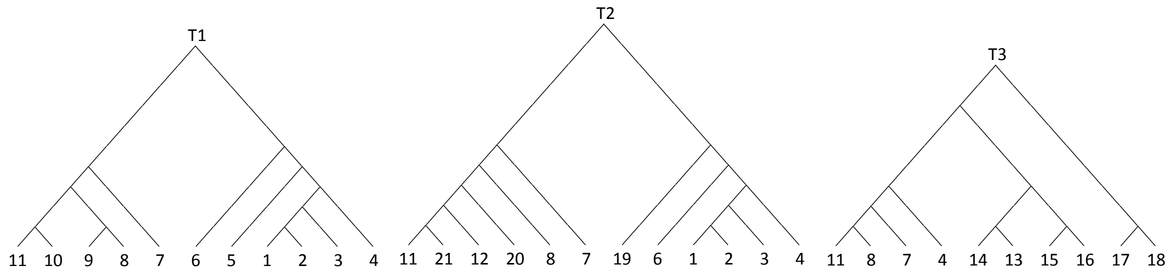


**Figure 4.13**: Assembly sequence trees for a family of three pilot valves (variants # 29-200, 68-410, and 66-200)

The consensus tree (master assembly sequence tree) generated by the developed MIP model is shown in Figure 4.14 (a). Unlike the illustrative example results, the objective function value for this data set is zero, which means that there is no tree dissimilarity between the master tree and any of the three trees and that the used assembly sequencing data was consistent. The assembly sequence tree shown in Figure 4.14 (b) was extracted for the new valve (variant # 66-300) from the master tree by considering components of the new valve only out of the 21 components included in the master tree. Since the new variant has a component (i.e. outer plug (22)), which did not appear in any of the three existing variants, the position of that component within the extracted assembly sequence should be then decided by the planner manually before finalizing the assembly sequence plan.

The advantage of the proposed model is best demonstrated through this case study by its ability to generate the assembly sequence for groups of components that never existed together in any of the studied variants such as the adjusting bolt (1), locking nut (2) and spring ring (3) which appeared in the first two variants only and the seal bolt (16), seat washer (17), and seat (18) which appeared in the third variant only; while, all the six components (1, 2, 3, 16, 17 and 18) exist in the new variant. Such an advantage cannot be achieved by simply retrieving the assembly sequence of the best matching existing variant to the new one.

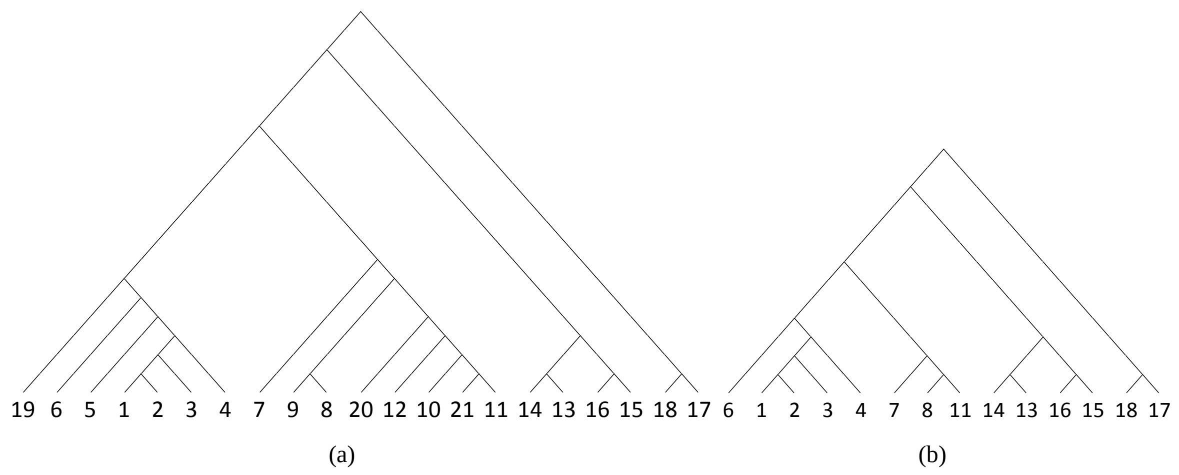


**Figure 4.14**: (a) Obtained master assembly sequence tree for a family of three control valves. (b) Assembly sequence tree for a new valve (variant # 66-300) extracted from the master assembly sequence tree

## 4.7 GA-Based Method for Constructing the Master Assembly Tree

The developed MIP model for constructing the master assembly tree is expected to be NP-hard (non-deterministic polynomial-hard) due to its strong combinatorial and binary nature. Accordingly, an alternative metaheuristic method, Based on a custom-designed Genetic Algorithm, for constructing the master assembly tree is developed. The metaheuristic method is made for problem sizes that could not be handled by the model in reasonable time. The method does not always guarantee that obtained master tree is the optimal one according to the defined objective function, especially for conflicting data; however, reasonable results could be still obtained in a reasonable time. MATLAB® programming and numerical computational software was utilized to implement the proposed GA-based method.

### 4.7.1 Methodology

Available assembly sequence trees are first encoded into a specially-designed *m* x *m* square matrix form (*m* is the number of leaves) to facilitate the GA operators to be presented later. The encoded matrix is also designed so that the corresponding tree could be restored easily. Consequently, for a given set of *N* trees and a total of *n* different components, there is an unknown *n* x *n* square matrix that represents the consensus (master) tree of all the available *N* trees. Hence, a Genetic Algorithm is applied to a set of initial randomly generated consensus trees (initial population) to search for the optimal consensus matrix that has the minimum sum of Robinson-Foulds distances from each of the individual matrices. Figure 4.15 shows an IDEF0 model of the proposed method illustrating the main activities as well as inputs, outputs, controls and mechanisms for each activity.

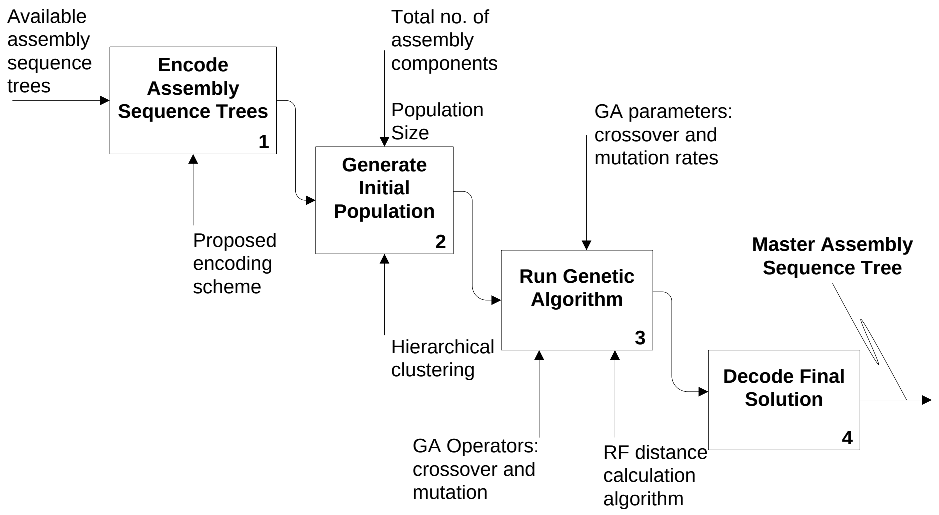

Available assembly sequence trees
Encode Assembly Sequence Trees
1
Proposed encoding scheme
Total no. of assembly components
Population Size
Generate Initial Population
2
Hierarchical clustering
GA parameters: crossover and mutation rates
Run Genetic Algorithm
3
GA Operators: crossover and mutation
RF distance calculation algorithm
Master Assembly Sequence Tree
Decode Final Solution
4

**Figure 4.15**: IDEF0 model for GA-based search for consensus trees

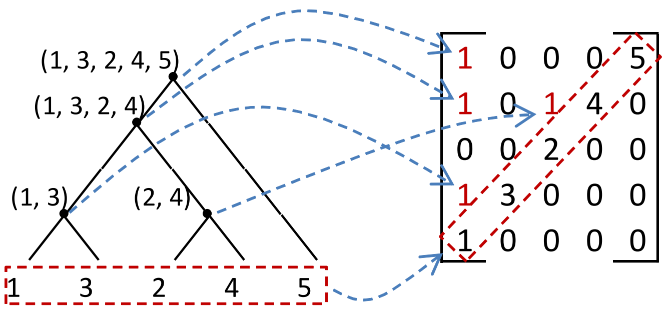


**Figure 4.16**: Proposed tree-to-matrix encoding scheme

### 4.7.2 Tree Encoding Scheme

A new tree-to-matrix encoding scheme, based on the one introduced by ElMaraghy and AlGeddawy (2012), is used to calculate the Robinson-Foulds distance between any two trees as well as to support Genetic Algorithm implementation. In encoding a tree of *m* leaves into a matrix, the information to be represented by the matrix is the hierarchy of the *m* elements which is equivalent to those elements belonging to the same node. This is carried out through the encoding scheme as shown in Figure 4.16.

Two types of tree information are encoded: *sequence of leaves* and *topology* or tree structure. Sequence of the leaves is encoded by the diagonal elements of the encoding matrix, while topology is encoded by the locations of binary (0-1) elements above the diagonal.

### 4.7.3 Generating Initial Population of Master Assembly Sequence Trees

A GA (Holland 1992; Godinho Filho et al. 2014) starts with an initial set of solutions known as initial population. This would be in the form of assembly sequence trees of *n* leaves or its equivalent matrices, where *n* is the total number of different leaves of all individual trees. To generate an *n* x *n* matrix, representing a feasible assembly sequence tree of *n* leaves, a set of *n* random coordinates are generated. An agglomerative (bottom up) hierarchical clustering algorithm, based on Euclidian distance as a proximity measure and single-linkage (nearest neighbor) as a clustering method (Jain et al. 1999), is applied to build the corresponding binary hierarchical clustering tree, also known as dendrogram. Dendrogram is an equivalent representation to assembly sequence trees used in this method.

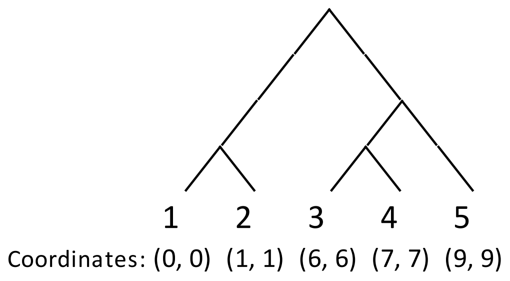


**Figure 4.17**: Tree of 5 leaves generated by hierarchical clustering

The “Linkage” function of the Statistics toolbox of MATLAB® was used for hierarchical clustering. The Linkage function output is converted into the proposed matrix form before proceeding to the Genetic Algorithm iterations. Figure 4.17 shows the tree obtained by hierarchical clustering if the coordinates (0, 0), (1, 1), (6, 6), (7, 7) and (9, 9) are randomly generated. Equal integer values of *x* and *y* coordinates are used here for simplicity, but in the actual algorithm implementation real numbers, which are not necessarily equal, are used to minimize the possibility of having ties in proximity which occurs when two or more minimum Euclidian distances between different clusters have the same value during the agglomerative process. Ties in proximity could be totally eliminated in single-linkage clustering by adding/subtracting a small value to/from the coordinates of leaves resulting in ties.

### 4.7.4 Fitness Function

Many algorithms have been developed for calculating the Robinson-Foulds distance with different computational efficiency in finding and storing the *S* sets (in Equation 4.2). Some of these algorithms are exact (Asano et al. 2010) and others are approximate (Pattengale et al. 2007). In this research, an algorithm based on the proposed matrix representation has been developed. Obtaining the *S* set for any given tree becomes rather straightforward given the used matrix encoding scheme. With reference to Figure 4.18, the subset of the *S* set that represents node *D* of the shown tree simply includes the group of consecutive diagonal elements {1, 3, 2, 4}, identified by the position of the cell representing *D* in the corresponding matrix.

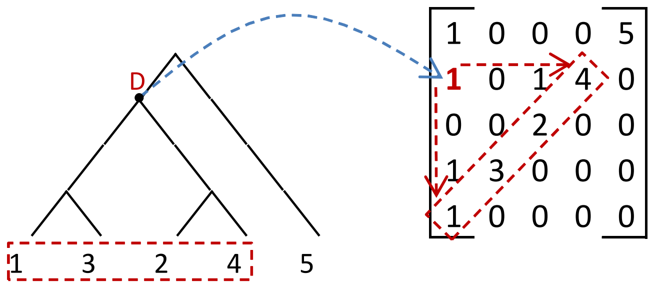


**Figure 4.18**: Obtaining the subsets of the C set for a given tree

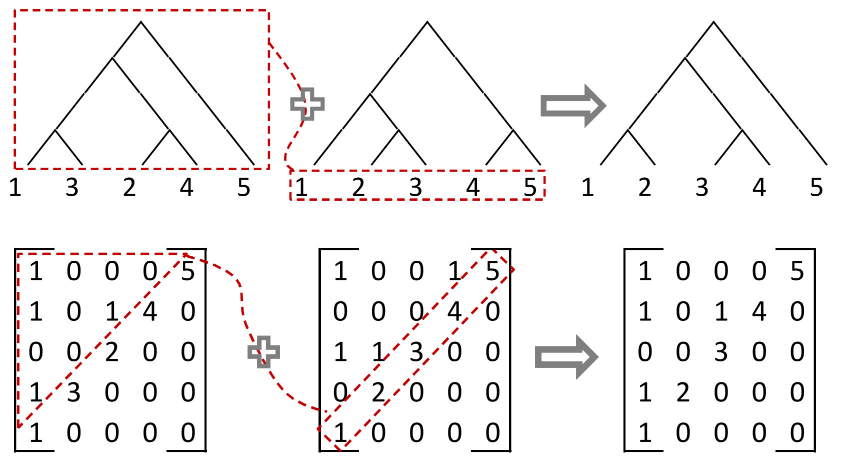
Throughout the proposed GA iterations, a fitness function is used to assess the fitness of any candidate solution (master assembly sequence tree). For a given set of available assembly sequence trees *N*, with a total of *n* different components, and a candidate master tree *MT*, the fitness function is the average of the Robinson-Foulds distances between the candidate master tree *MT* and every individual tree *T* out of the *N* available trees (Equation 4.16). In most cases, *MT* has more components than *T*. Thus, during calculating Robinson-Foulds distance between a given master tree *MT* and any individual tree *T*, all elements that exist in *MT* but not in *T* are ignored and temporarily removed from the *C* set of *MT*.

$$Fitness = \frac{\sum_{i=1}^{i=N} RF(MT, T_i)}{N} \quad (4.16)$$

### 4.7.5 Genetic Algorithm Operators

*Crossover*

A specially designed crossover operation is developed to allow proper information exchange between any two combined trees which guarantees the feasibility of the new tree. According to the proposed crossover operation, for two given randomly selected matrices, a new matrix is generated by obtaining (inheriting) the topology part of one matrix (elements above the diagonal) and the leaves sequence part of the other matrix (diagonal elements). Hence, there is no chance of generating matrices that are not equivalent to valid assembly sequence trees. Figure 4.19 illustrates the proposed crossover operator mechanism in both tree and matrix forms. The tree form is shown for clarification.



$$\begin{bmatrix}1&0&0&0&5\\1&0&1&4&0\\0&0&2&0&0\\1&3&0&0&0\\1&0&0&0&0\end{bmatrix} + \begin{bmatrix}1&0&0&1&5\\0&0&0&4&0\\1&1&3&0&0\\0&2&0&0&0\\1&0&0&0&0\end{bmatrix} \Rightarrow \begin{bmatrix}1&0&0&0&5\\1&0&1&4&0\\0&0&3&0&0\\1&2&0&0&0\\1&0&0&0&0\end{bmatrix}$$

**Figure 4.19**: Proposed crossover operator in tree and matrix forms

*Mutation*

Three mutation operators where developed; two of them involve mutating sequences of leaves (diagonal elements of the matrix form) and the third mutates topologies (elements above the diagonal in the matrix form). The first mutation operator reverses the sequence of leaves of a randomly selected tree. The second swaps two randomly selected leaves. The third randomly alters the topology of a randomly selected tree. As mentioned in Section 4.6.3, the initial population is produced by random generation of coordinates that are then clustered into trees through agglomerative hierarchical clustering. Hence, performing the topology mutation task without generating invalid trees could be simply done by assigning a new random coordinate to a randomly selected leaf of a randomly selected tree then rebuilding the tree. Figure 4.20 shows how the tree in Figure 4.17 would look like if the coordinate of the second leaf from the left (leaf 2) was changed from (1, 1) to (10, 10) without changing the sequence of the leaves. According to that perturbation, node *D* in the original tree disappears and new node *K* is formed instead in the mutated tree.

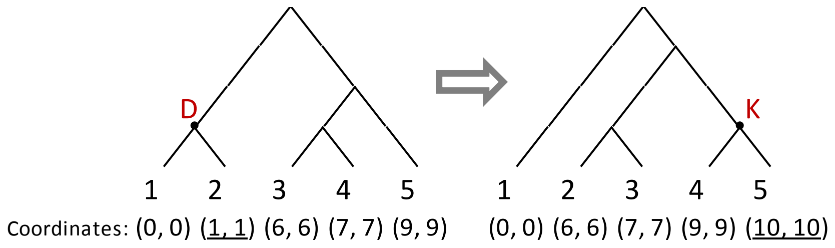


**Figure 4.20**: Proposed topology mutation operator

*Local Search*

A local search (LS) routine is applied, during each GA iteration, to the current global best solution in order to find better ones locally. Similar to the third mutation operator, three local search operations are carried out to examine minor altered solutions from the global best one so far. The first operation is to swap two randomly selected coordinates. The second is to alter the value of a randomly selected coordinate by adding or subtracting a randomly generated value within pre-defined limits. The third operation is to insert randomly selected coordinates in a randomly selected position. After each local search operation a new tree (topology and sequence) is built and assessed and the global best solution is updated accordingly. Each time any of these local search operations is applied it is kept iterating for a predefined number of iterations (e.g. 20 iterations for each local search operation). Furthermore, the global best solution is always kept in any newly generated population.

All the input needed for the proposed GA-based method is the encoded assembly sequence matrices. The method finds the master assembly matrix and then decodes it back into a tree form, in which the final output of the method is the master assembly tree. The results of the test runs have favored the following values for the GA parameters: 0.7 for crossover, 0.1 for each mutation operator, 25 for no. of local search trials and 300 for population size. It should also be mentioned that a perfect consensus tree (with a zero objective function) does not always exist, hence, the algorithm is designed to terminate if a perfect consensus tree is obtained or when it exceeds a pre-defined number of GA iterations without improving the objective function. Using the mentioned parameters values, The GA-based method was applied to the valves case study used to test the MIP model, in which the optimal master assembly tree was successfully obtained in less than quarter the time taken by the MIP model.

## 4.8 Summary

A novel approach has been introduced for knowledge-based assembly sequence planning which generates the master assembly sequence tree for a family of assembled products to be used for constructing the assembly sequence tree for a new product family member. The proposed approach is inspired by the concept of consensus trees found in evolutionary studies. Building the master assembly sequence tree is considered as a "clustering" type of application for knowledge discovery. For cases involving conflicting sequences (i.e. having multiple different assembly sequences for the same combination of components), the proposed method generates the master assembly tree following the most common sequences found for those conflicting components. Semi-final assembly sequence trees can be extracted for new variants that fall within, or significantly overlap with, the scope/boundary of the studied family of products. The model is capable of generating assembly sequences for products containing groups of components that did not exist together in any of the considered individual variants. This advantage of generating the master assembly sequence tree cannot be realized by retrieving the closest individual assembly sequence.

A Mixed-Integer Programming model was developed for constructing the master assembly tree. The model is NP-hard (non-deterministic polynomial-hard) due to its strong combinatorial and binary nature. Accordingly, an alternative GA-based method, with a custom designed crossover and mutation operators, was developed for cases that could not be handled by the MIP model in a reasonable time. Future work would go further with the concept of consensus trees and apply it to manufacturing/machining applications. Consequently, the sequence of machining operations for a new part could then be extracted based on the sequence of operations for similar existing/previous parts, even if the new part has a group of operations that did not exist together in any of the considered individual parts.

# CHAPTER 5
# MANUFACTURING SYSTEM SYNTHESIS BY ASSOCIATION RULES DISCOVERY

## 5.1 Introduction

The fourth product development application addressed in this research is manufacturing system synthesis. A systematic knowledge-based approach for synthesizing manufacturing system capabilities should speed up product development time and shorten manufacturing lead time. This chapter introduces a novel mathematical model for synthesizing manufacturing system capabilities for a new product design through the discovery of implicit knowledge accumulated over time in manufacturing data of historical products. Given the data on a set of existing or old products and the corresponding manufacturing system capabilities used in their manufacturing, an Integer Programming (IP) model is formulated and tested to discover associations between manufacturing capabilities and product features. This kind of discovered knowledge could be used to synthesis the required manufacturing capabilities for prospective products of new combinations of features.

Association rule discovery, one of the major knowledge discovery types, have many useful applications to product development and manufacturing (Kusiak 2006; Choudhary et al. 2009). The particular application studied in this research for association rule discovery has been recently addressed by AlGeddawy and ElMaraghy (2011; 2012). The model developed in that pioneering research is based on a tree-based classification method, known as cladistics, adopted from Biology and Phylogenetics (Page 2003).

The novel association rule discovery approach proposed in this chapter, based on Integer Programming, offers a more straightforward single-stage approach for this problem that requires significantly less implementation effort. The model is tested using the milling machines case study introduced by AlGeddawy and ElMaraghy (2011) as well as a new case study for engine block assembly.

## 5.2 Literature review

Association rule discovery, also known as association rule learning and association rule mining, is one of the typical knowledge discovery subjects. It is concerned with finding interesting correlations between data items in databases. Association rule discovery was popularized particularly due to the prominent work of Agrawal et al. (1993). Agrawal et al. studied the problem of mining a large collection of basket data transactions for significant association rules between sets of items (products). An example for an association rule could be a statement that “87% of customers purchasing bread and butter also purchase milk (i.e. {Eggs, Butter} $\rightarrow$ {Bread}), with 43% support. The 87% is the confidence of the rule; while, the support of a given rule is the proportion of transactions in the data set by which the rule occurred.

Agrawal et al. handled the problem as two sub-problems, the first and the main one is to generate all sets of items that have transaction support above a certain threshold (frequent itemsets). The second sub-problem, which is a straightforward one, is to derive rules satisfying a predefined minimum confidence between each of the frequent itemsets generated from solving the first sub-problem. Apriori algorithm (Agrawal and Srikant 1994) by Agrawal and Srikant is one of the very famous association rule discovery algorithms. Apriori could be considered as a refined or a mature version from the first algorithm of Agrawal et al. (1993). Several further association rule discovery algorithms have been then developed that are mainly seeking better computational efficiency (Stumme et al. 2002; Zaki 2004; Lakhal and Stumme 2005). A comprehensive survey for association rule discovery algorithms is found in (Ceglar and Roddick 2006). Many software packages that perform association rule discovery are currently available (e.g. Magnum Opus, TANAGRA and Weka). Researchers either use one of the available packages (Jiao et al. 2008), or build their own algorithm (Wang et al. 2005). Many industrial applications to association rule discovery do exist in literature. The following sub-sections review successful applications in three major fields: quality control, product design, and manufacturing and process control.

*Quality Control*

Chen et al. (2004) developed a manufacturing defect detection method based on the discovery of association rules between combinations of machines and the likehood of having defective products and applied it to a case study in the semiconductor industry. Da Cunha et al. (2006) studied the association between the assembly sequence and the likehood of having faulty products to be used for sequencing modules and forming product families that minimize the cost of production faults. Kusiak (2002) addressed the association between product quality and process control parameters and used a metal forming case study for demonstration. Buddhakulsomsiri et al. (2006) used association rule discovery to correlate product attributes and causes of failure in automotive industry using warranty data. Maki and Teranishi (2001) developed quality control system based on the discovery of association rules between the occurrence of faults and the machines used in the production process and applied it to an LCD fabrication example.

*Product Design*

Agard and Kusiak (2004) developed a product family design methodology based on association rules between various product options/features. Shao et al. (2006) addressed the product configuration design problem by studying association between clusters of product specifications and configuration alternatives. A case study of an electrically powered bicycle was used for illustration. Song and Kusiak (2009) developed an approach for optimizing product configuration based on discovered associations among the product options/features provided to the customers. The approach has been demonstrated on different automotive products. Jiao and Zhang (2005) developed a methodology for product portfolio identification through the generation of association rules between customer needs and product specifications and applied it to a case study for a vibration motor.

*Manufacturing and Process Control*

Jiao et al. (2008) proposed a process planning support tool based on association rule discovery for correlations between product features and process elements and applied it to a case study of vibration motors for cell phones. Sodyan et al. (2006) developed an algorithm for manufacturing process control based on association rules between the process input and output parameters and demonstrated it through an industrial example of spray forming process, which is a rapid tool making technology that adopts rapid prototyping. Mahamaneerat et al. (2007) developed a cell-formation method that groups machines based on discovered association rules between them through production records. Wang et al. (2005) proposed a process planning support tool based on generated association rules between manufacturing requirements and the parts complexity.

Other industrial applications for association rule discovery include job shop scheduling (Yan and Shi 2013), facility layout (Altuntas et al. 2013) and order batching (Azadnia et al. 2013). The particular association rule discovery problem studied in this research has been recently addressed by AlGeddawy and ElMaraghy (2011) using a novel non-statistical approach. AlGeddawy and ElMarghy developed a co-evolution model for manufacturing systems and products, inspired by theories of natural species co-evolution, to eventually discover relationships between product features and needed manufacturing capabilities.  In that model, the manufacturing capabilities of each system and the set of features of each product variant produced by those systems are first binary encoded in matrix forms. Then, both systems and products are classified using a data clustering technique called cladistics that generates classification trees called cladograms. The classification trees of systems and products are then reconciled (matched) to obtain the best similar pair of trees. Corresponding branches of systems and products trees provide relationships between manufacturing capabilities and product features. Figure

5.1 shows an example of using cladistic analysis to discover associations between manufacturing system capabilities and product features.

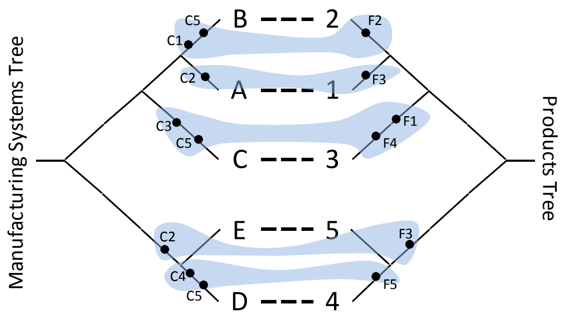


**Figure 5.1**: Association rule discovery using cladistic analysis by (figure reproduced from (AlGeddawy and ElMaraghy 2012))

Association rule discovery algorithms based on the work of Agrawal et al. (1993), which is the typical tool used by most practitioners in the field, generates very large number of redundant rules in which a lot of post processing work is needed to interpret meaningful rules out of the large number of generated rules. Furthermore, properly defining minimum statistical thresholds is a critical decision and the obtained results are quite sensitive to these thresholds. Compared to the non-statistical co-evolution model of AlGeddawy and ElMaraghy (2011), the developed model is more straightforward and easier to implement. This is a new solution approach to the problem that could be easily carried out in one step through mathematical programming, in which there is no need to build trees then match them and finally establish association which is a three-step approach.

## 5.3 Problem Description and Research Scope

This research introduces a novel mathematical programming model for discovering association relationships between manufacturing system capabilities (manufacturing domain) and product features (product domain). The developed model is not limited to manufacturing-product domains only; it can deal as well with assembly-product domains. In general, the model is capable of extracting associations between any paired groups of data (e.g. input and output parameters of a given manufacturing process). However, the model is not appropriate for subjective types of data such as data based on customer preferences, in which significant inconsistency or discrepancy can be found.

An example of a capability in the context of this research could be the type of material handling equipment (e.g. Automated Guided Vehicle (AGV) or robot arm) or the type of machine tool (e.g. milling or turning). While, an example of a product feature could be the geometrical features (e.g. rotational or prismatic) or the components name or code in case of assembly products (e.g. frame variant x or spring variant y). The Opitz product classification and coding system (Opitz 1970) is one method for classifying part features. A novel and unique method for coding and classifying manufacturing systems modules (machines, material handling equipment, etc.) and their layout was first introduced by ElMaraghy (2005) and further developed and applied in (ElMaraghy et al. 2010; Samy and ElMaraghy 2012).

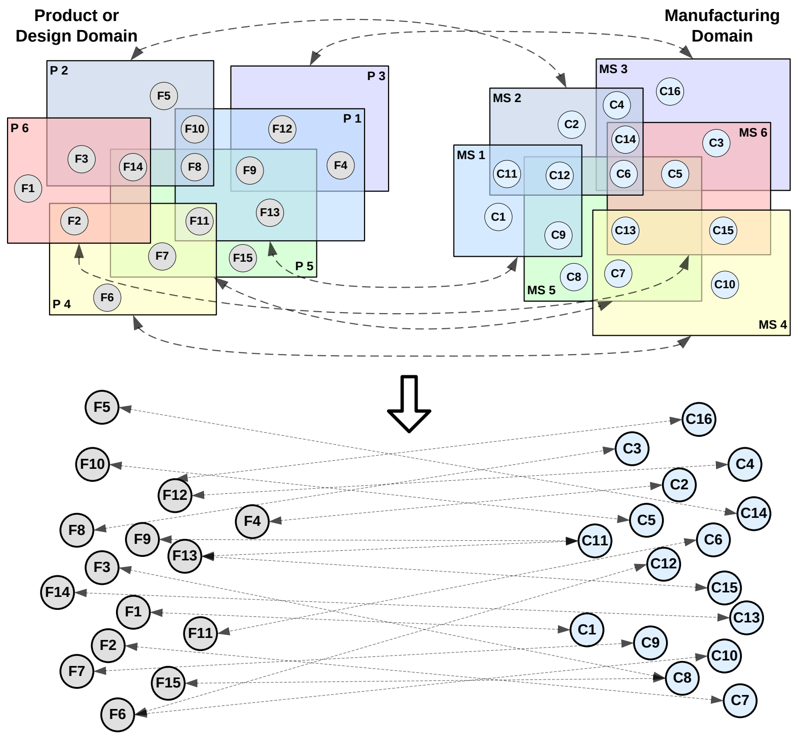


**Figure 5.2**: Schematic description for the studied problem

The problem addressed in this chapter is defined as follows: given a number of old and/or existing manufacturing data instances where each instance involves data about a given product (P) and the corresponding manufacturing system (MS) used to produce it, it is required to extract associations rules in terms of what manufacturing system capabilities are associated with what features and vice versa. In other words, as it is schematically described in Figure 5.2, it is required to construct a mapping between the product design domain, in terms of product features (F), and its manufacturing domain, in terms of manufacturing capabilities (C). In reality, the data needed for this knowledge discovery application could be taken from process sheets described before in Chapter 4. Process sheets in fact include detailed data about machines, equipment, fixtures and toolings used to process or assemble any given product. For large companies, such data could be extracted from PLM (Product Lifecycle Management) systems which are the systems integrating tools such as CAD (Computer-Aided design), CAM (Computer-Aided Manufacturing) and PDM (Product Data Management).

It should be noted that an association between a given manufacturing capability C and a product feature F does not mean that C by itself is capable of making F. For example, if an association relationship is found between a given type of fixtures and a given type of geometric feature, this certainly does not mean that such a fixture can solely guarantee achieving the specification of that feature. In fact, the usefulness or applicability of association relationships is best realized in combination. Hence, when a new combination of product features arises, the corresponding manufacturing system capabilities that should be able to produce that product is synthesized by combining all manufacturing capabilities associated with every feature in the new product.

Throughout the coming sections, the main focus and terminology will be on manufacturing (mainly machining) while Section 5.7 will show how to extend the use of the proposed model to the assembly application.

## 5.4 Association Rule Discovery Integer Programming Model

The proposed mathematical model seeks to find/discover associations between manufacturing system capabilities and product features. It is an Integer Programming model that has three main sets of constraints. The model is inspired by the *set covering* problem; a classical problem in Computer Science and Complexity theory (Beasley and Chu 1996). Given a number of tasks to be performed, and the group of alternative resources that can perform each task, the set covering problem therefore finds the least number of resources that can perform all tasks. The proposed model seeks to find the least number of association rules that satisfy all given manufacturing data instances. Applications that can be modeled as set covering problem include facility location, airline crew scheduling, assembly line balancing, and vehicle routing problems (Gouwanda and Ponnambalam 2008).

Unlike other association rule discovery methods, except for (AlGeddawy and ElMaraghy 2011), this model does not look for the top ranked association rules that satisfy some minimum statistical support measures. Every data instance is considered by the model as a manufacturing solution that has been successfully applied before and hence the model considers the entire sets of rules that satisfy every instance of the given data set. The manufacturing data to be studied by the proposed model are first encoded in a binary matrix from. As shown in Table 5.1, two groups of data are involved; a group for manufacturing system capabilities and a corresponding group of product features. Each instance of available data is represented by one line that encodes the set of manufacturing capabilities and the product features it has actually produced.

**Table 5.1**: An example for the input data to the proposed model

| Manuf. System | Manufacturing Capabilities | | | | Product | Product Features | | | |
|---|---|---|---|---|---|---|---|---|---|
| | C1 | C2 | … | Cv | | F1 | F2 | … | Fu |
| MS 1 | 1 | 0 | | 1 | Prod. 1 | 1 | 0 | | 0 |
| MS 2 | 0 | 0 | | 1 | Prod. 2 | 1 | 1 | | 0 |
| : | | | | | : | | | | |
| MS 3 | 1 | 1 | | 1 | Prod. n | 0 | 0 | | 1 |

The parameters of the proposed model are stated as follows:

| | |
|---|---|
| $s$ | Total number of data instances (products or systems) |
| $u$ | Number of product features |
| $v$ | Number of manufacturing system capabilities. |
| $c_{ik}$ | A binary (0-1) element that takes value of "1" if capability $k$ exists in manufacturing system $i$, otherwise it is "0" (e.g. an element in the left hand side of Table 5.1) |
| $f_{ij}$ | A binary (0-1) element that takes value of "1" if product feature $j$ exists in product $i$, otherwise it is "0" (e.g. an element of the right hand side of Table 5.1) |
| $b_{km}$ | A binary (0-1) element that takes value "1" if the two capabilities $k$ and $m$ should not be associated together with the same feature. |

**Table 5.2**: An example for the association matrix obtainable by the proposed model

| | | Product Feature (F) | | | | |
|---|---|---|---|---|---|---|
| | | F1 | F2 | F3 | F4 | F5 |
| Capability (C) | C1 | 1 | 0 | 0 | 0 | 1 |
| | C2 | 0 | 0 | 0 | 1 | 0 |
| | C3 | 1 | 0 | 1 | 0 | 0 |
| | C4 | 0 | 0 | 0 | 0 | 1 |
| | C5 | 0 | 1 | 0 | 0 | 0 |
| | C6 | 1 | 1 | 0 | 0 | 0 |

The proposed model has one main set of decision variables $a$, where $a_{kj}$ is a binary element in the matrix shown in Table 5.2 (association matrix); $a_{kj}$ takes the value "1" if the manufacturing capability $k$ is associated with product feature $j$ and "0" otherwise. Table 5.2 shows an example of an association matrix for a data set that involves six manufacturing capabilities and five product features. The objective function (Equation 5.1) of the proposed model is to minimize the total sum of the $a$ variables (association matrix). This is corresponding to finding the least number of association rules which are sufficient to explain the given data. Finding the optimal values of $a$ (association matrix) is the ultimate objective of the model.

$$\text{Min} \sum_{k=1}^{v} \sum_{j=1}^{u} a_{kj} \quad (5.1)$$

In addition to the binary constraint (Equation 5.5) applied to the association matrix, the model employs three main sets of constraints. The first two (Equations 5.2 and 5.3) ensure that the values to be assigned by the model for each $a$ variable do satisfy the relations between each product feature and manufacturing capability as given by the data. That is to say, if the resulting $a$ matrix is used along with the already available product features set of data, the corresponding manufacturing capabilities set of data can be obtained and vice versa (i.e. re-generation of data).

$$\sum_{k=1}^{v} c_{ik}\, a_{kj} \ \geq \ f_{ij} \qquad \begin{array}{l} i = 1, \dots, s \\ j = 1, \dots, u \end{array} \quad (5.2)$$

$$\sum_{j=1}^{u} f_{ij}\, a_{kj} \ \geq \ c_{ik} \qquad \begin{array}{l} i = 1, \dots, s \\ k = 1, \dots, v \end{array} \quad (5.3)$$

The third set of constraints (Equation 5.4) prevents any two manufacturing capabilities that are mutually exclusive or even two capabilities that either of them can include the other, from being associated together with the same feature. This is how the model recognizes alternatives of the same type of capability.

$$a_{kj} + a_{mj} b_{km} \leq 1 \quad \begin{array}{l} k = 1, \dots, v \\ m = 1, \dots, v;\ m > k \\ j = 1, \dots, u \end{array} \quad (5.4)$$

Table 5.3 shows an example of the matrix used to define manufacturing capabilities that should not be associated together with the same product feature ($b$ matrix). In this example, C1 and C2 are two mutually exclusive alternatives of a same type of manufacturing capability, thus they form one group of capabilities and they should not be jointly associated with the same product feature. C3 and C4 in this example also form one group of manufacturing capabilities.

**Table 5.3**: Matrix defining alternatives of same type of capabilities

| | | Capability (C) | | | | |
|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | C4 | C5 |
| Capability (C) | C1 | 0 | 1 | 0 | 0 | 0 |
| | C2 | 0 | 0 | 0 | 0 | 0 |
| | C3 | 0 | 0 | 0 | 1 | 0 |
| | C4 | 0 | 0 | 0 | 0 | 0 |
| | C5 | 0 | 0 | 0 | 0 | 0 |

Table 5.4 shows an example where two manufacturing capabilities of the same type are accompanied with the same product feature. In this example, the model, without the third constraint, has two

equivalent solutions for this set of manufacturing data; the first is to associate F3 with C1 and C2 and to associate F1 with C3 and the second is to associate F3 with C2 and C3 and to associate F1 with C1. Both solutions will lead to the same number of association rules. However, with the third constraint the second choice, which is the more sensible solution is only feasible and is thus enforced by the model. If it happened to have a feature that could be associated with two or more capabilities of the same type, an association rule is generated only for the relation that minimizes the objective function (i.e. the more frequently found relation).

**Table 5.4**: An example showing importance of the third constraint

| Machine | Manufacturing Capabilities | | | Product | Product Features | | |
|---|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | | F1 | F2 | F3 |
| M/C 1 | 1 | 0 | 1 | Prod. 1 | 1 | 0 | 1 |
| M/C 2 | 0 | 1 | 0 | Prod. 2 | 0 | 0 | 1 |

This model is developed to deal with real data. In some cases, the input data may be conflicting or inconsistent, in which case a feasible solution that satisfies all constraints cannot be obtained. For instance, C1 and C2 in the data shown in Table 5.5 both belong to the same group. Although this situation is not likely to occur, this data does not lead to a feasible solution. The only solution for this example is to associate C1, C2 and C3 with F1 which is not allowed by the third constraint which prevents any two manufacturing capabilities that are mutually exclusive or even two capabilities that either of them can include the other, from being associated together with the same feature (Equation 5.4).

**Table 5.5**: An example showing importance of the auxiliary decision variables

| Machine | Manufacturing Capabilities | | | Product | Product Features | | |
|---|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | | F1 | F2 | F3 |
| M/C 1 | 1 | 0 | 1 | Prod. 1 | 1 | 0 | 0 |
| M/C 2 | 0 | 1 | 0 | Prod. 2 | 1 | 0 | 0 |

Therefore, two sets of auxiliary decision variables *h* and *g* are introduced to the model to relax (when necessary) the first and the second constraints, respectively, in order to always guarantee feasible solutions for any kind of data. The sum of each of those two variables is also added to the objective function to minimize their values. Furthermore, a weighted objective function is used in order to ensure the higher priority of the original term.  Accordingly, *h* and *g* are assigned non-zero values by the model only when inconsistent data such as the one shown in Table 5.5 is involved. The modified form of the first two constraints is given by Equations 5.2' and 5.3' and the corresponding modified objective function is given by Equation 5.1'.

$$\text{Minimize} \sum_{k=1}^{v}\sum_{j=1}^{u} a_{kj} + \sum_{i=1}^{s}\sum_{j=1}^{u} 10h_{ij} + \sum_{i=1}^{s}\sum_{k=1}^{v} 10g_{ik} \qquad (5.1')$$

$$\sum_{k=1}^{v} c_{ik}\, a_{kj} + h_{ij} \ \geq \ f_{ij} \qquad i = 1,\dots,s;\ j = 1,\dots,u \qquad (5.2')$$

$$\sum_{j=1}^{u} f_{ij}\, x_{kj} + g_{ik} \ \geq \ c_{ik} \qquad i = 1,\dots,s;\ k = 1,\dots,v \qquad (5.3')$$

The complete proposed knowledge discovery Integer Programming model, with non-negativity constraints on *h* and *g* variables (Equations 5.6 and 5.7), is formulated as follows:

$$\text{Minimize} \sum_{k=1}^{v}\sum_{j=1}^{u} a_{kj} + \sum_{i=1}^{s}\sum_{j=1}^{u} 10h_{ij} + \sum_{i=1}^{s}\sum_{k=1}^{v} 10g_{ik} \qquad (5.1')$$

*Subject to:*

$$\sum_{k=1}^{v} c_{ik}\, a_{kj} + h_{ij} \ \geq \ f_{ij} \qquad i = 1,\dots,s;\ j = 1,\dots,u \qquad (5.2')$$

$$\sum_{j=1}^{u} f_{ij}\, a_{kj} + g_{ik} \ \geq \ c_{ik} \qquad i = 1,\dots,s;\ k = 1,\dots,v \qquad (5.3')$$

$$a_{kj} + a_{mj}b_{km} \leq 1 \qquad k = 1,\dots,v;\ m = 1,\dots,u;\ m > k;\ j = 1,\dots,u \qquad (5.4)$$

$$a_{kj} \in \{0,1\} \qquad k = 1,\dots,v;\ j = 1,\dots,u \qquad (5.5)$$

$$h_{ij} \geq 0 \qquad i = 1,\dots,s;\ j = 1,\dots,u \qquad (5.6)$$

$$g_{ik} \ \geq 0 \qquad i = 1,\dots,s;\ k = 1,\dots,v \qquad (5.7)$$

## 5.5 Validation Example

This section presents a simple example to illustrate and validate the developed Integer Programming model and highlight its differences from existing alternative methods. The model is validated by generating data based on already known association rules, and comparing those rules with the discovered ones using the IP model. Consider the manufacturing data instances given in Table 5.6, where four manufacturing capabilities and five product features are involved. C3 and C4 are two alternatives of the same type of capability. This data was actually generated using the association matrix given in Table 5.7. Arbitrary product features were assumed at each row of Table 6 and the corresponding capabilities were obtained from Table 5.7. Thus, Table 5.7 is considered as certain (actual) data to be later compared with the results of the model.

**Table 5.6:** Manufacturing data instances for the verification example

| Machine | Manufacturing Capabilities | | | | Product | Product Features | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | C4 | | F1 | F2 | F3 | F4 | F5 |
| M/C 1 | 1 | 1 | 1 | 0 | Prod. 1 | 1 | 1 | 0 | 0 | 0 |
| M/C 2 | 1 | 1 | 0 | 0 | Prod. 2 | 1 | 0 | 0 | 0 | 1 |
| M/C 3 | 1 | 0 | 0 | 1 | Prod. 3 | 0 | 0 | 1 | 1 | 0 |
| M/C 4 | 0 | 1 | 1 | 0 | Prod. 4 | 0 | 1 | 0 | 0 | 0 |
| M/C 5 | 0 | 1 | 0 | 1 | Prod. 5 | 0 | 0 | 1 | 0 | 1 |
| M/C 6 | 1 | 1 | 0 | 1 | Prod. 6 | 1 | 0 | 1 | 0 | 0 |

**Table 5.7:** The actual relations from which the manufacturing data was generated

| | | Product Feature (F) | | | | |
|---|---|---|---|---|---|---|
| | | F1 | F2 | F3 | F4 | F5 |
| Capability (C) | C1 | 1 | 0 | 0 | 1 | 0 |
| | C2 | 1 | 1 | 0 | 0 | 1 |
| | C3 | 0 | 1 | 0 | 0 | 0 |
| | C4 | 0 | 0 | 1 | 0 | 0 |

Table 5.8 shows the matrix defining groups of alternative manufacturing capabilities (*b* matrix). The model was written and solved using the LINGO optimization software package. The model was able to obtain an association matrix that exactly matches the actual data in Table 5.7. Consistent data instances were employed in this example and thus, all *h* and *g* auxiliary variables have zero values.

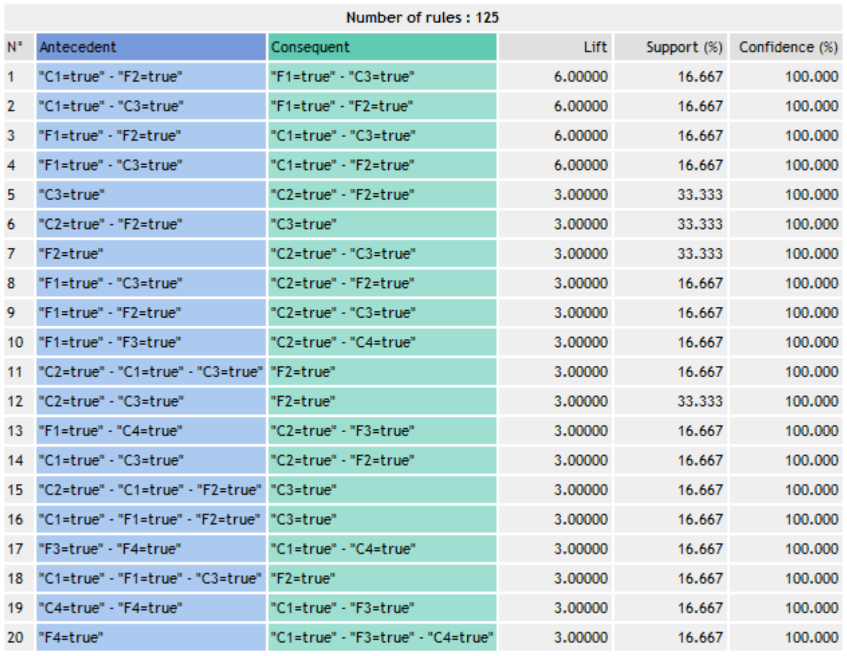
**Table 5.8:** Matrix defining groups of alternative capabilities

| | | Capability (C) | | | |
|---|---|---|---|---|---|
| | | C1 | C2 | C3 | C4 |
| Capability (C) | C1 | 0 | 0 | 0 | 0 |
| | C2 | 0 | 0 | 0 | 0 |
| | C3 | 0 | 0 | 0 | 1 |
| | C4 | 0 | 0 | 0 | 0 |

Although, the model reached 100% matching solution in this example, the number of data instances as well as the degree by which the instances vary from each other can affect the model results; larger number of instances and more different instances allow better recognition of association relationships. For example, if the last data instance was removed from Table 5.6, an association matrix which does not perfectly match the actual one is obtained as shown in Table 5.9. Accordingly, one relation (with darker background) is missing. However, even if this incomplete association matrix is used in conjunction with the product data side of Table 5.6 (first five rows to the right); the same corresponding data for the manufacturing system side (first five rows to the left) could still be re-generated, conforming to the second constraint of the model. In general, in order to get reliable results, reasonable number of data instances - with respect to the number of capabilities and features - as well as reasonable distinction between them, is in fact needed.

**Table 5.9:** Association matrix obtained by the model if the last data instance was removed

| | | Product Feature (F) | | | | |
|---|---|---|---|---|---|---|
| | | F1 | F2 | F3 | F4 | F5 |
| Capability (C) | C1 | 1 | 0 | 0 | 1 | 0 |
| | C2 | 0 | 1 | 0 | 0 | 1 |
| | C3 | 0 | 1 | 0 | 0 | 0 |
| | C4 | 0 | 0 | 1 | 0 | 0 |



Number of rules : 125

| N° | Antecedent | Consequent | Lift | Support (%) | Confidence (%) |
|---|---|---|---|---|---|
| 1 | "C1=true" - "F2=true" | "F1=true" - "C3=true" | 6.00000 | 16.667 | 100.000 |
| 2 | "C1=true" - "C3=true" | "F1=true" - "F2=true" | 6.00000 | 16.667 | 100.000 |
| 3 | "F1=true" - "F2=true" | "C1=true" - "C3=true" | 6.00000 | 16.667 | 100.000 |
| 4 | "F1=true" - "C3=true" | "C1=true" - "F2=true" | 6.00000 | 16.667 | 100.000 |
| 5 | "C3=true" | "C2=true" - "F2=true" | 3.00000 | 33.333 | 100.000 |
| 6 | "C2=true" - "F2=true" | "C3=true" | 3.00000 | 33.333 | 100.000 |
| 7 | "F2=true" | "C2=true" - "C3=true" | 3.00000 | 33.333 | 100.000 |
| 8 | "F1=true" - "C3=true" | "C2=true" - "F2=true" | 3.00000 | 16.667 | 100.000 |
| 9 | "F1=true" - "F2=true" | "C2=true" - "C3=true" | 3.00000 | 16.667 | 100.000 |
| 10 | "F1=true" - "F3=true" | "C2=true" - "C4=true" | 3.00000 | 16.667 | 100.000 |
| 11 | "C2=true" - "C1=true" - "C3=true" | "F2=true" | 3.00000 | 16.667 | 100.000 |
| 12 | "C2=true" - "C3=true" | "F2=true" | 3.00000 | 33.333 | 100.000 |
| 13 | "F1=true" - "C4=true" | "C2=true" - "F3=true" | 3.00000 | 16.667 | 100.000 |
| 14 | "C1=true" - "C3=true" | "C2=true" - "F2=true" | 3.00000 | 16.667 | 100.000 |
| 15 | "C2=true" - "C1=true" - "F2=true" | "C3=true" | 3.00000 | 16.667 | 100.000 |
| 16 | "C1=true" - "F1=true" - "F2=true" | "C3=true" | 3.00000 | 16.667 | 100.000 |
| 17 | "F3=true" - "F4=true" | "C1=true" - "C4=true" | 3.00000 | 16.667 | 100.000 |
| 18 | "C1=true" - "F1=true" - "C3=true" | "F2=true" | 3.00000 | 16.667 | 100.000 |
| 19 | "C4=true" - "F4=true" | "C1=true" - "F3=true" | 3.00000 | 16.667 | 100.000 |
| 20 | "F4=true" | "C1=true" - "F3=true" - "C4=true" | 3.00000 | 16.667 | 100.000 |

**Figure 5.3**: A snap shot from TANAGRA showing 20 association rules obtained using Apriori ("Lift" is an additional statistical measure)

To better highlight the difference between the developed Integer Programming model and other existing methods, the results obtained using a typical association rule discovery software, as well as the results obtained using the cladistics-based model of AlGeddawy and ElMaraghy (2011) are compared. Figure 5.3 exhibits a snapshot showing a partial view of the results obtained by TANAGRA data mining software package (http://eric.univ-lyon2.fr/~ricco/tanagra/). In order to get a comparable type of outcome to that obtained by the developed model, rules with 100% confidence only, regardless of their support values, were generated using the Apriori algorithm of TANAGRA. Selecting proper support and confidence levels is critical as it can significantly impact the quality of generated rules.

As shown in Figure 5.3, the algorithm provided all possible association rules even those correlating elements of the same group of data (e.g. rule no. 5 in Figure 3: C3 → C2 and F2). Nevertheless, many rules that are dominated by other rules are also provided as both rules may have 100% confidence but at different levels of support (e.g. rules no. 11 and 12). In addition, unlike the results obtainable by the developed model, the Apriori rules are one way rules, where one side of the rule leads to the other side, while the opposite is not true. Consequently, a very large number of redundant rules are generated; the number of association rules obtained for this small example is 125, the snapshot in Figure 5.3 only shows 20 of them. Thus, this method is more appropriate for discovering association rules in applications featuring discrepancies and inconsistency such as market-based data where statistical significance measures are essential.

As for the co-evolution model of AlGeddawy and ElMaraghy (2011), Figure 5.4 shows the two best matching classification trees (cladograms). The tree on the left is one of the optimal (most parsimonous (Page 1994)) classification trees for the six products in this example and the tree on the left is one of the optimal classification trees for the six corresponding manufacturing systems. The features (or capabilities) included in any given product (or system) are given by the features (or capabilities) on all the branches that connect the tree root to the product (or system) itself at the tree leaf. The TNT cladistic analysis software (http://www.cladistics.com/) was utilized for generating these trees.

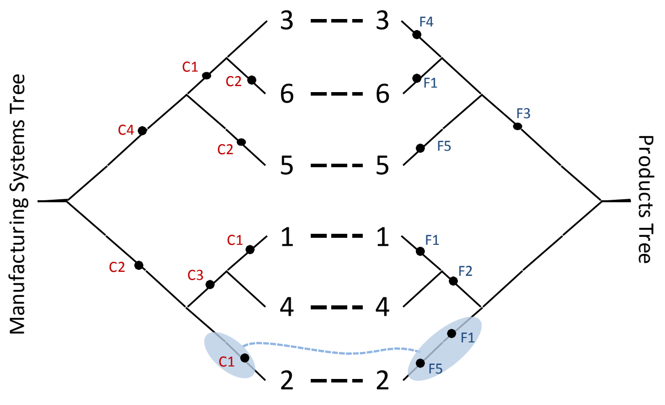


**Figure 5.4**: Matched most parsimony trees for products and systems

The association between features (F) and capabilities (C) is obtained by correlating features and capabilities which lie on every two corresponding branches. Each row in Table 5.10 expresses an association rule between a feature or group of features and a capability or group of capabilities. For instance, the association between C1 on one side and F1 and F5 on another side is given by the eighth row of the Table. This is the association rule highlighted in the trees shown in Figure 5.4. Table 5.10 can be further re-formatted to obtain Table 5.11 in the same outcome format of the developed Integer Programming model. Two of the expected association rules which were not generated, and one unexpected association rule was generated; these are highlighted in red in Table 5.11.

**Table 5.10:** Association rules obtained from matched trees

| C1 | C2 | C3 | C4 | F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 5.11:** The final association rules aggregating all relations of Table 5.10

| | | Product Feature (F) | | | | |
|---|---|---|---|---|---|---|
| | | F1 | F2 | F3 | F4 | F5 |
| Capability (C) | C1 | 1 | 0 | 0 | 0 | 1 |
| | C2 | 1 | 0 | 0 | 0 | 1 |
| | C3 | 0 | 1 | 0 | 0 | 0 |
| | C4 | 0 | 0 | 1 | 0 | 0 |

Generating an unexpected relation is considered discovery of an alternative relation to a missing one. However, the number of discovered rules is still less than number needed to fully describe (i.e. re-generate) the given data. Finding, all rules needed to re-generate the given data is in fact a different approach to association rule discovery that is adopted by the proposed IP model. This is a more conservative approach to association rule discovery that may result in some cases in more than the needed set of capabilities for a given combination of features in which the user has to select from. On the other hand, the cladistic-based method discovers the minimum core and strongly associated capabilities needed to produce a given feature. Therefore, using the cladistic-based method the user may occasionally need to select additional capabilities which were not suggested by the model. The developed Integer Programming association rule discovery model is easier to implement than the cladistic-based model.

In conclusion, the developed IP model is superior to conventional association rule discovery algorithms that look for all association rules satisfying some predefined statistical support thresholds.

These methods, in addition to being quite sensitive to the used statistical thresholds, they generate a huge number of redundant rules that require extensive filtering to arrive at valid and useful rules. In many cases more than 95% of the rules would need to be excluded. As shown in this example, 125 rules were generated while in fact 7 rules only were needed to fully describe the association relationships in the given data. Compared to the cladistic-based model, the IP model is different solution methodology which requires less implementation and development effort. When using the proposed IP model, there is no need to build trees then match them and finally build associations which is a three steps approach. Furthermore, the IP model is more automated as it provides the association results directly in a matrix form while for instance, in the cladistic-based method, associating capabilities and features on matching branches of the cladogram is done by the user. A practitioner only needs to be familiar with IP programming to implement the proposed model.

## 5.6 Case Study I – Machining

In this section, the developed association rule discovery Integer Programming model is applied to the case study addressed in (AlGeddawy and ElMaraghy 2011). The manufacturing system used in this case study includes seven milling machines and the products are parts machined using them (Figure 5.5). Seven manufacturing instances are considered, each of them represents one of the machines and the corresponding part it is capable of producing.

### 5.6.1 Case Study Data

The input data for the case study are shown in Tables 5.12 and 5.13. The seven machines are characterized by six groups of manufacturing capabilities with a total of eleven capabilities as follows:

1. Structure: Horizontal/Vertical
2. Axes of Motion: 3 Axes/4 Axes/ 5 Axes
3. No. of Machining Heads: One/Two
4. Turning Spindle
5. Turret/Tool Magazine
6. Control: Manual/CNC

The machined parts are characterized by six groups of features with a total of twelve features as follows:

1. Dimensionality: Rail/Cube/Rectangular cross section
2. Shape: Rectangular/Non-Rectangular/Compound Block
3. Rotational Features
4. Machined Surfaces: One Direction/Stepped Surfaces from one direction/More Direction

**5.** Special Surfaces: Keyways or Grooves/Complex Surfaces
**6.** Auxiliary Holes

The coding format for both the machines and product features are somewhat different here than it has originally appeared in (AlGeddawy and ElMaraghy 2011). For instance, the first product feature "Structure" is originally coded as a single feature in which a value of "1" for that feature means that the structure is horizontal while a value of "0" means that the structure is vertical. It is more appropriate to split that feature into two separate sub-features so as to allow identification of explicit association rules for each alternative of that features. All alternative features and capabilities are encoded here as separate features or capabilities except for on/off features or capabilities that may either exist or not.

**Table 5.12**: Seven milling machines and their manufacturing capabilities

| **Machine** | **Manufacturing Capabilities** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1 Structure** | | **2 Axes of Motion** | | | **3 No. of Heads** | | **4** | **5** | **6 Control** | |
| | Vertical | Horizontal | 3 Axes | 4 Axes | 5 Axes | One | Two | **Turning Spindle** | **Turret/Tool Magazine** | Manual | CNC |
| **1** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **2** | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **4** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **5** | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **6** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| **7** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

**Table 5.13**: Corresponding machined parts and their features

| **Part** | **Product Features** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1 Dimentio-nality** | | **2 Shape** | | | **3** | **4 Machined Surface** | | | **5 Special Surface** | | **6** |
| | Rail | Cube | Rectangular | Non Rec. | Compound | **Rotational Feature** | One Dir. | Stepped | More | Keyway | Complex | **Auxiliary Holes** |
| **1** | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| **3** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **4** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **5** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| **6** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **7** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

Tables 5.12 and 5.13 should both be read together as if they are one table of two sides; for instance, machine 1 from Table 5.6 with capabilities: “Structure: Horizontal”, “Axes of Motion: 3-Axes”, No. of Heads: One”, no “Turning Spindle”, no “Turret/Tool Magazine”, and “Control: Manual” was used to produce part 1 from Table 5.7 with “Dimensionality: Cube”, “Shape: Rectangular”, no “Rotational Features”, “Machined Surface: One Direction”, no “Special Surfaces”, and no “Auxiliary Holes”. More details about the description of the capabilities and features in Tables 5.12 and 5.13 are found in the original publication from which the case study was obtained. The manufacturing system capabilities considered in this case study is limited to the direct machine capabilities, however further capabilities with regard to the used fixtures and tooling could be also considered.

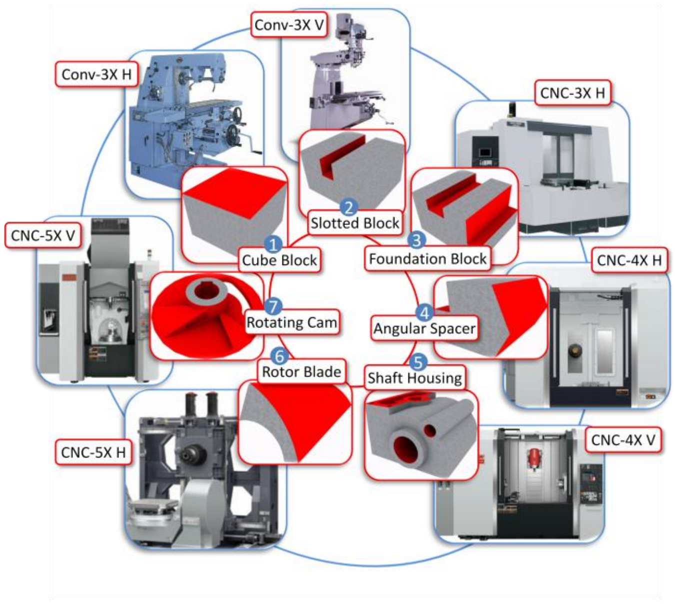


**Figure 5.5**: The studied machines and their corresponding parts (modified from (AlGeddawy and ElMaraghy 2012))

**Table 5.14**: The matrix defining groups of manufacturing capabilities

| | | **Manufacturing Capabilities** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | | **2** | | | **3** | | **4** | **5** | **6** | |
| **Manufacturing Capabilities** | **1** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **2** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **3** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The corresponding matrix (*b* matrix) which defines alternatives of the same type of manufacturing capability is given by Table 5.14, where a cell with a value "1" means that the two manufacturing capabilities corresponding to that cell are alternatives of the same type of capability and should not be associated together with the same feature. For instance, cell (1, 2) has a value of "1" because any given feature should be only associated to machine structure that is either horizontal or vertical. If it happened to have a feature that has been assigned in the data to two or more capabilities of the same type, then the association rule will be generated for the relation that minimizes the objective function (i.e. the more frequent relation).

**Table 5.15**: Association matrix obtained by the proposed Integer Programming model

| | | | **Product Features** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **1. Dimentionality** | | **2. Shape** | | | **3. Rotational Feature** | **4. Machined Surface** | | | **5. Special Surfaces** | | **6. Auxiliary Holes** |
| | | | Rail | Cube | Rectangular | Non Rec. | Compound | | One Dir. | Stepped | More | Keyway | Complex | |
| **Manufacturing Capabilities** | **1. Structure** | Vertical | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | Horizontal | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **2. Axes of Motion** | 3 Axes | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 4 Axes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | | 5 Axes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | **3. No. of Heads** | One | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | Two | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | **4. Turning Spindle** | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **5. Turret/Tool Mgazine** | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | **6. Control** | Manual | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | CNC | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 5.6.2 Results and Discussion

Seventeen association rules have been generated by the developed model. The auxiliary variables *h* and *g* of the model were not assigned any values, which mean that the data is consistent. The association matrix, which includes all rules, is provided in Table 5.15. Each highlighted cell in Table 5.15 represents an association rule between its corresponding feature and capability. For instance, cell (2, 2) represents an association between the feature of "Dimensionality: Cube" and the capability of "Structure: Horizontal". Accordingly, the capability "Structure: Horizontal" should be among the manufacturing capabilities required for producing the "Dimensionality: Cube" feature. The knowledge provided in the discovered association matrix could also be outlined in a different form, where each feature is listed, along with the manufacturing capabilities associated with it as shown in Table 5.16. An arrow in Table 5.16 refers to required manufacturing capability (or capabilities) by the corresponding features. Hence, for a given part of a new combination of features, the corresponding required manufacturing capabilities can be synthesized using the discovered rules. For example, if it is required to manufacture a new part with the following set of features:

**Table 5.16**: Manufacturing capabilities associated with each product feature

| | | | |
|---|---|---|---|
| **1**.Dimentionality: | Rail | → | • Control: CNC |
| | Cube | → | • Structure: Horizontal<br>• Axes of Motion: 3 Axes<br>• Turning Spindle |
| **2**. Shape: | Rectangular | → | • Control: Manual |
| | Non-Rectangular | → | • Structure: Horizontal |
| | Compound Block | → | • Control: CNC |
| **3**. Rotational Feature: | | → | • Structure: Vertical |
| **4**. Machined Surface: | One Direction | → | • No of Heads: One |
| | Stepped Surface | → | • No of Heads: One<br>• Turret/Tool Magazine |
| | More Direction | → | • Axes of Motion: 4 Axes<br>• No of Heads: One |
| **5.** Special Surfaces: | Keyways/Grooves | → | • Structure: Vertical |
| | Complex Surfaces | → | • Axes of Motion: 5 Axes<br>• No of Heads: Two |
| **6**. Auxiliary Holes | | → | • Axes of Motion: 4 Axes |

- Dimensionality: Rail
- Shape: Non-Rectangular
- Machined Surface: Stepped
- Auxiliary Holes

Then according to the discovered association rules in Table 5.15 or Table 5.16, a machine that at least has following set of manufacturing capabilities is needed:

- Structure: Horizontal
- No of Heads: One
- Turret/Tool Magazine
- Axes of Motion: 4 Axes
- Control: CNC

None of the existing machines has this combination of features. The closet existing machine to this combination of features is machine number 4 which only lacks the turret/tool magazine capability associated with the stepped machined surface feature. Hence, the manufactures may either modify this machine and equip it with a turret/tool magazine, or buy a new machine that has all the identified capabilities including a tool magazine.

The developed Integer Programming model for this example was solved using LINGO in less than one second on a PC of 2.8 GHz Quad Core processor and 4 GB RAM.  Due to the limited number of data instances relative to number of capabilities and features, this case study does not have a unique solution. That is to say, there are several other association matrices that can satisfy the model constraints with 17 rules. For that reason, there is no sufficient basis for conducting a detailed comparison between the association rules obtained by the developed Integer Programming model and the rules produced using the cladistic-based model as reported in (AlGeddawy and ElMaraghy 2011).

The obtained results shows the ability of the model to discover the implicit manufacturing patterns by extracting the relationships between milling machines' structural components and capabilities and the geometrical features of the machined parts. Some postulated instances of newly introduced parts and machines were used to demonstrate the use of the discovered relationships for synthesizing new manufacturing systems for the new products or even finding applications for new machines.

The developed association rule discovery model is not limited to manufacturing systems/products data. It is also capable of extracting association rules between assembly system components and their corresponding assembly products as to be shown in the next section. Furthermore, the model is applicable to other paired groups of data such as input and output parameters of a given manufacturing process. The work of Sodyan et al. (2006) on manufacturing process control is one example for further applications for the proposed model beside manufacturing/assembly system synthesis.

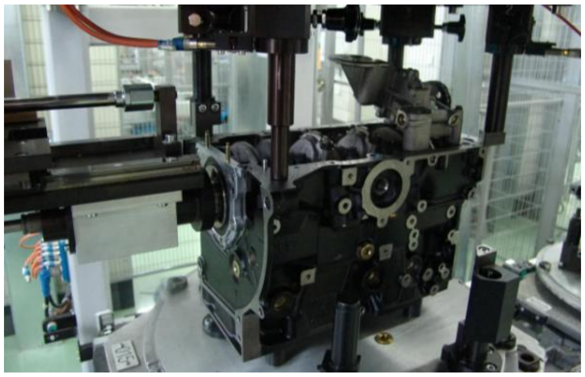
## 5.7 Case Study II - Assembly

This section presents an industrial case study of engine block assembly - particularly the short block sub-assembly - that demonstrates the applicability of the developed association rule discovery model to assembly system synthesis, in addition to its applicability to machining system synthesis. A short block is the sub-assembly of the automobile engine block below the head gasket. Major parts of a short block sub-assembly are the cylinder block, pistons, connecting rods, crank shaft, crank shaft cap, oil pump, oil pump chain, and oil pan. It also contains many other less significant parts - mainly standard parts - such as bushings, pins, bolts, nuts, rings, and so on. Assembly operations for a short block are either manual, semi-automated (in which powered assembly equipment or tools are used to assist the operator) or totally automated.

The main assembly operations for short blocks are essentially similar for various engine models. Hence, unlike the previous example which was concerned with system capabilities level, the assembly system synthesis in this example will be mainly concerned with the level of "*stations setup*", which is basically the type of assembly operation as well as the used assembly equipment, tools, fixtures, etc.

Different assembly station setups are required for different engine block models depending on the material by which the cylinder block is made of as well as the size and shape of the engine. For instance, aluminum cylinder blocks require piston liners while steel blocks do not. Different engine sizes also mean different sizes and perhaps geometries of the major components. Hence, different tools, fixtures and assembly equipment settings would be needed for different engine models.

Accordingly, it would be beneficial to extract association relationships between short block components and the assembly station setups to expedite the process of assembly planning and system set-up when a new engine model is introduced. The data available for this example is for two 4-cylinder inline engine models that the company used to produce before at different periods (Figure 5.5). Further real images for the short block assembly to be presented in this section are obtained through personal communication (Kourosh Khedri, December 1, 2014). Additional seven data instances are generated based on the original two instances. For confidentiality reasons, company information will not be revealed and altered engine model acronyms are used.


**Engine model: TW3-EZX**

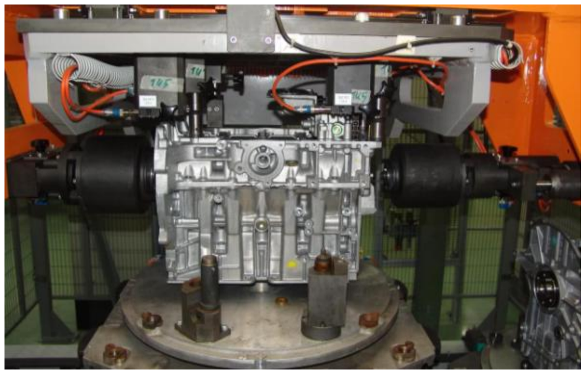

**Engine model: TW5-ABA**

**Figure 5.5:** Real image for two different engine block models

**Table 5.17:** Nine engine variants and their parts

| **Part No.** | **Part Name** | **Part Code** | **Engine Variant** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **TW3-EZX** | **TW3-EZY** | **TW3-EZV** | **TW3-EZN** | **TW3-EZM** | **TW5-ABA** | **TW5-ABB** | **TW5-ABM** | **TW5-ABE** |
| **1** | Cylinder Block | **CB** | CB-V1 | CB-V2 | CB-V3 | CB-V3 | CB-V3 | CB-V4 | CB-V5 | CB-V4 | CB-V5 |
| **2** | Cylinder Liner | **CL** | n/a | n/a | n/a | n/a | n/a | CL-V1 | CL-V2 | CL-V1 | CL-V2 |
| **3** | Piston | **PN** | PN-V1 | PN-V1 | PN-V2 | PN-V2 | PN-V1 | PN-V3 | PN-V3 | PN-V3 | PN-V4 |
| **4** | Connecting Rod | **CR** | CR-V1 | CR-V1 | CR-V2 | CR-V2 | CR-V1 | CR-V3 | CR-V3 | CR-V3 | CR-V4 |
| **5** | Crank Shaft | **CS** | CS-V1 | CS-V1 | CS-V2 | CS-V2 | CS-V2 | CS-V2 | CS-V4 | CS-V3 | CS-V4 |
| **6** | Oil Pump | **OP** | OP-V1 | OP-V1 | OP-V3 | OP-V2 | OP-V2 | OP-V3 | OP-V4 | OP-V4 | OP-V4 |
| **7** | Middle Case | **MC** | MC-V1 | MC-V2 | MC-V3 | MC-V3 | MC-V3 | n/a | n/a | n/a | n/a |
| **8** | Crank Shaft Cap | **CSC** | n/a | n/a | n/a | n/a | n/a | CSC-V1 | CSC-V2 | CSC-V1 | CSC-V2 |
| **9** | Oil Pump Chain | **OPC** | OPC-V1 | OPC-V2 | OPC-V3 | OPC-V3 | OPC-V3 | OPC-V4 | OPC-V5 | OPC-V4 | OPC-V5 |
| **10** | Oil Pan | **OPN** | OP-V1 | OP-V2 | OP-V3 | OP-V2 | OP-V2 | OP-V3 | OP-V4 | OP-V3 | OP-V3 |

**Table 5.18:** Nine engine variants and their assembly station setups

| **Station No.** | **Description** | **Level of Automation** | **Station Code** | **Engine Variant** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | **TW3-EZX** | **TW3-EZY** | **TW3-EZV** | **TW3-EZN** | **TW3-EZM** | **TW5-ABA** | **TW5-ABB** | **TW5-ABM** | **TW5-ABE** |
| **1** | Load cylinder block to pallets | semi-automated | **ST1** | ST1-S1 | ST1-S2 | ST1-S3 | ST1-S3 | ST1-S3 | ST1-S4 | ST1-S5 | ST1-S4 | ST1-S5 |
| **2** | Load pistons and connecting rods to pallets | manual | **ST2** | ST2-S1 | ST2-S1 | ST2-S2 | ST2-S2 | ST2-S1 | ST2-S3 | ST2-S3 | ST2-S3 | ST2-S4 |
| **3** | Assemble connecting rods to the pistons | Semi-automated | **ST3** | ST3-S1 | ST3-S1 | ST3-S2 | ST3-S2 | ST3-S1 | ST3-S3 | ST3-S3 | ST3-S3 | ST3-S4 |
| **4** | Loosen connecting rods bolts | semi-automated | **ST4** | ST4-S1 | ST4-S1 | ST4-S2 | ST4-S2 | ST4-S1 | ST4-S3 | ST4-S3 | ST4-S3 | ST4-S4 |
| **5** | Install piston rings | manual | **ST5** | ST5-S1 | ST5-S1 | ST5-S3 | ST5-S3 | ST5-S2 | ST5-S4 | ST5-S5 | ST5-S5 | ST5-S5 |
| **6** | Insert piston-rod sub-assembly into liners | semi-automated | **ST6** | n/a | n/a | n/a | n/a | n/a | ST6-S1 | ST6-S1 | ST6-S1 | ST6-S2 |
| **7** | Insert piston-rod sub-assembly into c. block | semi-automated | **ST7** | ST7-S1 | ST7-S1 | ST7-S1 | ST7-S2 | ST7-S2 | n/a | n/a | n/a | n/a |
| **8** | Insert liners inside the cylinder block | manual | **ST8** | n/a | n/a | n/a | n/a | n/a | ST8-S1 | ST8-S1 | ST8-S1 | ST8-S1 |
| **9** | Detach middle case from the cylinder block | semi-automated | **ST9** | ST9-S1 | ST9-S1 | ST9-S2 | ST9-S2 | ST9-S1 | n/a | n/a | n/a | n/a |
| **10** | Detach crankshaft caps | semi-automated | **ST10** | n/a | n/a | n/a | n/a | n/a | ST10-S1 | ST10-S2 | ST10-S1 | ST10-S2 |
| **11** | Detach connecting rod caps | manual | **ST11** | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 | ST12-S1 |
| **12** | Install crankshaft in the cylinder block | semi-automated | **ST12** | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 | ST14-S1 |
| **13** | Install and fasten the connecting rod caps | semi-automated | **ST13** | ST18-S1 | ST18-S2 | ST18-S1 | ST18-S2 | ST18-S2 | ST18-S1 | ST18-S3 | ST18-S2 | ST18-S3 |
| **14** | Installing chains for oil pumps | manual | **ST14** | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 | ST19-S1 |
| **15** | Deposit silicon for assembling middle case | automated | **ST15** | ST20-S1 | ST20-S2 | ST20-S3 | ST20-S3 | ST20-S3 | n/a | n/a | n/a | n/a |
| **16** | Install and fasten screws for middle case | automated | **ST16** | ST23-S1 | ST23-S2 | ST23-S3 | ST23-S3 | ST23-S3 | n/a | n/a | n/a | n/a |
| **17** | Install and fasten screws for crankshaft caps | automated | **ST17** | n/a | n/a | n/a | n/a | n/a | ST24-S1 | ST24-S2 | ST24-S2 | ST24-S1 |
| **18** | Install oil pump | manual | **ST18** | ST25-S1 | ST25-S1 | ST25-S3 | ST25-S2 | ST25-S2 | ST25-S3 | ST25-S4 | ST25-S4 | ST25-S4 |
| **19** | Fasten screws for oil pump | automated | **ST19** | ST26-S1 | ST26-S1 | ST26-S3 | ST26-S2 | ST26-S2 | ST26-S3 | ST26-S4 | ST26-S4 | ST26-S4 |
| **20** | Install crankshaft seals | automated | **ST20** | ST27-S1 | ST27-S1 | ST27-S2 | ST27-S2 | ST27-S2 | ST27-S2 | ST27-S4 | ST27-S3 | ST27-S4 |
| **21** | Deposit silicon sealant for oil pan | automated | **ST21** | ST28-S1 | ST28-S2 | ST28-S3 | ST28-S2 | ST28-S2 | ST28-S3 | ST28-S4 | ST28-S3 | ST28-S3 |
| **22** | Install oil pan | manual | **ST22** | ST29-S1 | ST29-S2 | ST29-S3 | ST29-S2 | ST29-S2 | ST29-S3 | ST29-S3 | ST29-S4 | ST29-S3 |
| **23** | Install and fasten screws for oil pan | semi-automated | **ST23** | ST30-S1 | ST30-S2 | ST30-S3 | ST30-S2 | ST30-S2 | ST30-S3 | ST30-S3 | ST30-S3 | ST30-S4 |
| **24** | Unload the short block assembly | automated | **ST24** | ST31-S1 | ST31-S1 | ST31-S1 | ST31-S1 | ST31-S1 | ST31-S2 | ST31-S2 | ST31-S2 | ST31-S2 |

The same steps followed in the machining example are also followed in this example. The differences are regarding the way by which products and systems are encoded. For the product side of the data, the short block parts are used as product features to describe and encode different short block variants. Ten different parts are considered for that. As for the system side of the data, there are 24 assembly stations; each assembly system is represented by the various stations and their setup. Table 5.17 shows the product data representation and Table 5.18 shows the corresponding assembly system data representation. The setup of a given station is represented by the symbol "S" followed by the setup number; for instance, "ST4-S3" would stand for station 4 with setup number 3. It should be mentioned that the level of details considered in this example for both the products and the systems are a little simplified than the reality for demonstration purposes. A schematic layout of the assembly system for one of the engine variants is shown in Figure 5.6. The layout would not be significantly different for other short block variants.

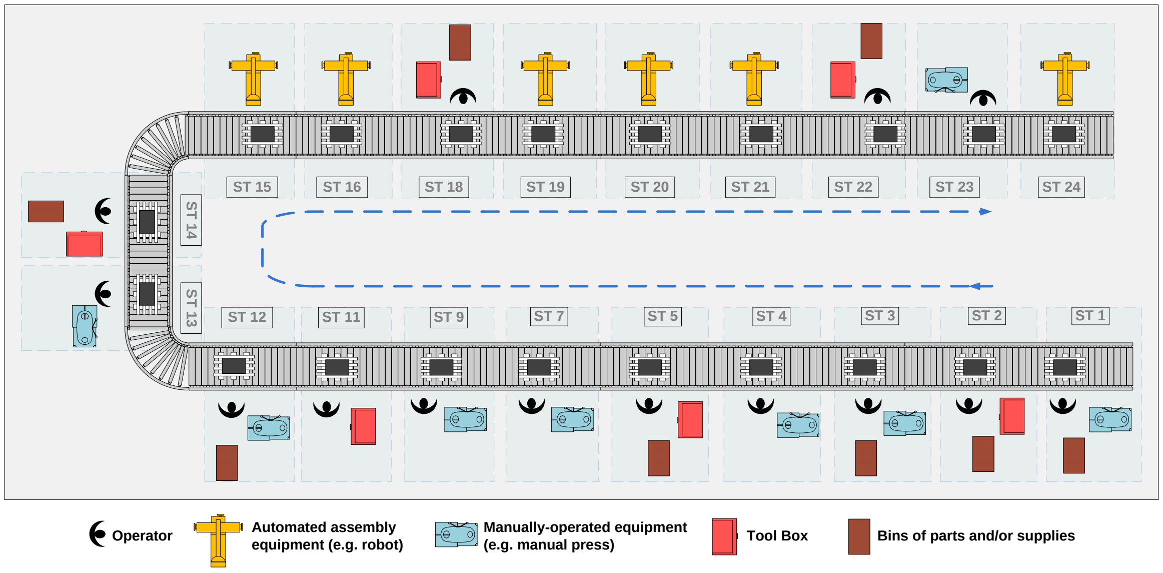

ST 15
ST 16
ST 18
ST 19
ST 20
ST 21
ST 22
ST 23
ST 24
ST 14
ST 13
ST 12
ST 11
ST 9
ST 7
ST 5
ST 4
ST 3
ST 2
ST 1
Operator
Automated assembly equipment (e.g. robot)
Manually-operated equipment (e.g. manual press)
Tool Box
Bins of parts and/or supplies

**Figure 5.6:** Schematic layout for the TW3-EZX engine model

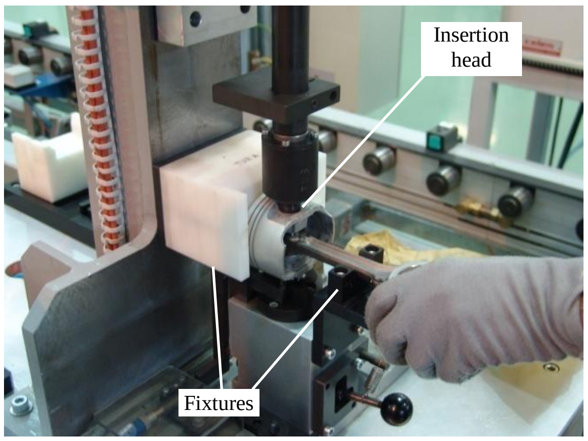


**Figure 5.7:** Assembling piston to the connecting rod at station 3

As an example for variations between different setups for the same station, the assembly operation performed at station 3 is to assemble connecting rod to the piston in which a manual press is used to insert the pin that connects the rod to the piston. Engine sizes vary and accordingly the sizes of the piston-rod sub-assembly are different. Hence, a different insertion head for the manual press, as well as different fixtures for the piston and the connecting rod would be needed (see Figure 5.7). Other stations such as station 6 (Figure 5.8) where the piston-rod sub-assembly is inserted into the cylinder liner do also have the same reason for having different setups.

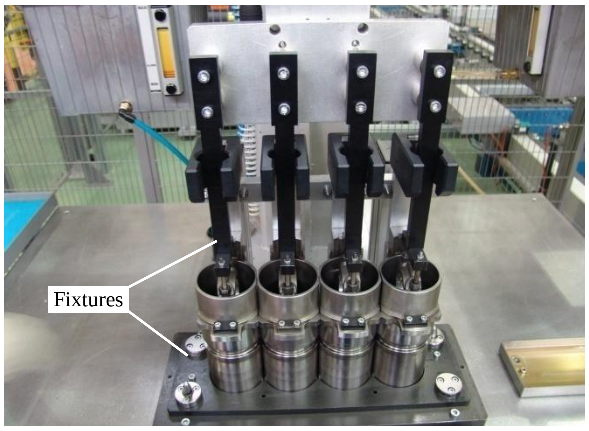


**Figure 5.8:** Insertion of the piston-rod sub-assembly into the cylinder liners at station 6

Another example for differences between setups of the same stations is the fastening of the crankshaft caps done at station 17. Pneumatic equipment is used to tighten all the crankshaft cap nuts simultaneously as shown in Figure 5.9. Setups for the used assembly equipment vary for different engine models in which the distances between the screwing nuts as well as the nuts sizes are different. Accordingly, different tightening heads and machine setups are needed. The same reason by which the setup of this station (station 17) varies also applies to similar stations by which there are screws to be tightened using pneumatically-powered equipment such as station 23 where the screws of the oil pan are tightened.

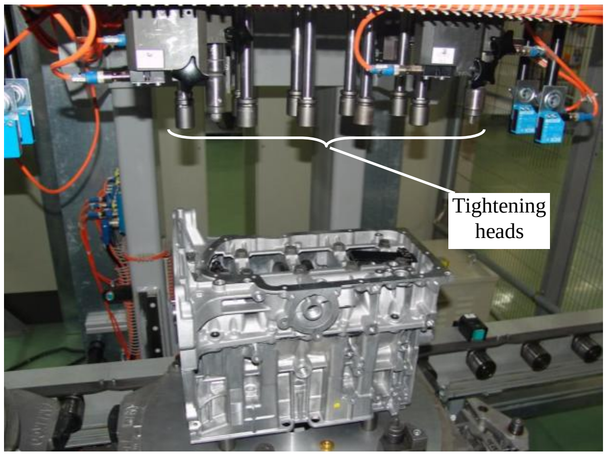


**Figure 5.9:** Fastening of the crankshaft caps at station 17

One more example is the need to change the setup for the automated assembly equipment used in station 21 to place silicon sealant for sealing the oil pan (Figure 5.10). A different path for the nozzle is required depending on the geometry and size of the oil pan. Different specification for the used silicon sealant may also be needed depending on the cylinder block and oil pan materials.

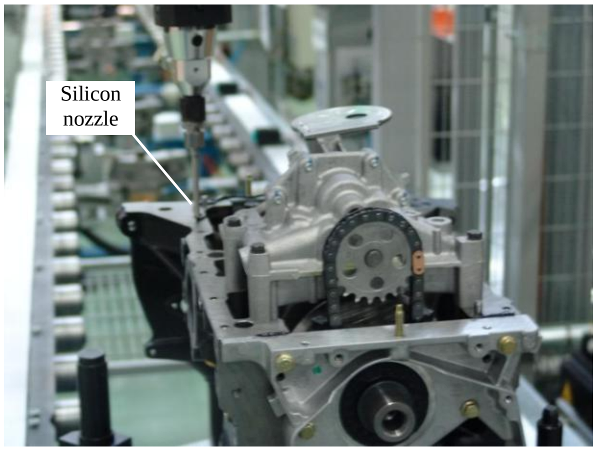


**Figure 5.10:** Deposit of silicon for oil pan seal at station 21

The corresponding binary-coded format for Tables 5.17 and 5.18 needed for the IP association rules discovery model is given in Tables A1 and A2, respectively, in Appendix A. The obtained association matrix with 83 rules between product features (engine block part variants) and assembly system components (station setups) is shown in Table A3 in Appendix A. Hence, for a given new engine model with a new short block design that shares a considerable number of part variants with the existing engine models, the obtained association rules would indicate exactly the required assembly stations as well as the setup for each station. For instance, for a new short block with following combination of part variants:

| | |
|---|---|
| 1. Cylinder Block: | CB-V**2** |
| 2. Cylinder Liner: | CL-V**2** |
| 3. Piston: | PN-V**3** |
| 4. Connecting Rod: | CR-V**3** |
| 5. Crank Shaft: | CS-V**4** |
| 6. Oil Pump: | OP-V**4** |
| 7. Middle Case: | n/a |
| 8. Crank Shaft Cap: | CSC-V**2** |
| 9. Oil Pump Chain: | OPC-V**1** |
| 10.Oil Pan: | OPN-V**2** |

The corresponding preliminary assembly system according to the obtained association results in Table A3 should be as follow:

| | | |
|---|---|---|
| ST1-S**2** | ST9-n/a | ST17-S**2** |
| ST2-S**3**/S**4** | ST10-S**2** | ST18-S**4** |
| ST3-S**3** | ST11-S**1** | ST19-S**4** |
| ST4-S**3** | ST12-S**1** | ST20-S**4** |
| ST5-S**5** | ST13-S**2** | ST21-S**2** |
| ST6-S**1** | ST14-S**1** | ST22-S**1**/S**2** |
| ST7-n/a | ST15-n/a | ST23-S**2**/S**3**/S**4** |
| ST8-S**1** | ST16-n/a | ST24-S**2** |

The synthesized system uses 20 stations out of the 24 available stations, since the association matrix does not recommend having stations 7, 9, 15 and 16 which are not needed for a variant that does not have the part “middle case”.

It is possible to obtain multiple alternatives for the same assembly system component by the proposed IP model. This is mainly attributed to the conservative nature of the model that seeks the association rules that completely satisfy the input data instances. In many cases, some of the these alternatives would be more advanced than the other alternatives, one simple way by which the designer can resolve such an issue would be to directly choose the more advanced alternative as it most likely contain the less advanced alternative. The use of more consistent data (i.e. the use of the same assembly system components for the same product variants), as well as having larger numbers of data instances should increase the discrimination ability of the model and hence minimize the likelihood of having multiple solutions.

## 5.8 Summary

An Integer Programming model is introduced, for the first time, to discover association rules between product features and manufacturing system capabilities. The proposed model is inspired by the classical set covering problem. The model identifies all association rules that are just sufficient to explain any given set of manufacturing data instances. This is a totally different approach from the conventional association rule discovery algorithms which look for all association rules satisfying some predefined statistical support thresholds. These methods, in addition of being quite sensitive to the used statistical thresholds, they generate huge number of redundant rules that require extensive filtering to arrive at valid and useful rules. Compared to the cladistic-based model of AlGeddawy and

ElMaraghy (AlGeddawy and ElMaraghy 2011), the developed model, from an implementation point of view, requires significantly less implementation and development effort.

The developed Integer Programming model is applicable to manufacturing as well as assembly applications. The model is in general capable of extracting association rules between any paired groups of data (e.g. input and output parameters of a given manufacturing process). However, the model is not recommended for subjective types of data such as data based on market surveys, in which significant data inconsistency/discrepancy often exists.

The discovered association rules by the model can be used as a manufacturing/assembly system synthesis support tool for future products with new combinations of features. This should greatly assist in speeding-up product development and systems design and re-design as new product features and variants are introduced. Size and quality of the available data is the main factor affecting the quality of the discovered knowledge rules. Future work may include determining the best set of found association rules, in cases where multiple solutions exist.

# CHAPTER 6
DISCUSSION AND CONCLUSIONS

## 6.1 Research Significance

This research has introduced a package of innovative knowledge discovery solutions to support different product development activities. Three main design/manufacturing applications were addressed: *product design retrieval*, *assembly sequence planning* and *manufacturing system synthesis.* Another application*, product family formation,* has been also addressed as an extension for BOM trees matching approach used for product design retrieval. Each of the developed methods in this research is based on a clear set of systematic steps that make them easy to implement and automate.

For the design phase of the product development process, a novel product design retrieval method based on BOM trees matching was developed. Once the closest matching old product design to a new one has been identified, many useful data previously generated for such a design can be easily extracted, modified and used for the new product design with obvious productivity benefits. These include CAD models, materials, tolerances as well as non-design related data such as assembly process plans, programs for assembly machinery and supply chain data.

As an extension for BOM trees matching approach, a new product family formation method, based on a novel Bill of Material (BOM) trees similarity measure, which addresses both components commonality and hierarchical structure of products, was developed. From an assembly point of view, grouping based on an assembly related similarity measure such as BOM trees should produce more reliable family formation results, in which sets of product families that share significant assembly system resources and setups are obtained, maximizing system utilization and productivity and decreasing system complexity. The method is systematic and easy to implement and automate.

For the assembly planning phase, a new approach for knowledge-based assembly sequencing was introduced, based on the notion of the master assembly sequence tree, to maximize the use of existing assembly sequence data in planning for new products. The model is capable of generating assembly sequences for products containing groups of components that did not exist together in any of the considered individual variants. This greatly simplifies and enhances automatic assembly sequence generation and minimizes subsequent modifications, hence, reducing assembly planning cost and improving productivity.

For the manufacturing system design (i.e. equipment selection) phase of the product development process, a practical mathematical model for discovering association rules between product design domain and manufacturing/assembly system domains was developed. Discovered rules could be used as a manufacturing support tool to synthesize manufacturing/assembly systems for future products of new combination of design features. This should greatly assist in speeding-up the manufacturing system design for future products. The proposed Integer Programming model is not limited only to system/product data and should be further capable of extracting association rules between other paired groups of data (e.g. input and output parameters of a given manufacturing process).

## 6.2 Engineering Thesis Questions

At this point of the thesis and after all the proposed models along with their results and significances have been introduced, it could be confidently said that the thesis of this research has been proven. That’s to say: *Innovative and well-designed knowledge discovery models could simplify and enhance the process of extracting useful patterns and knowledge from existing and historical product design and manufacturing data.*

Furthermore, it would be useful to conclude with the answer of the four typical engineering thesis questions as follow.

**1. The engineering problem to be solved**

The challenges faced by designers and manufacturers to enhance their design and manufacturing capabilities responsively and cost-effectively are greater than ever. Meanwhile, embedded design and manufacturing knowledge is accumulated over generations of product variants, based on designers and manufacturers experience. A great support to designers and manufacturers to cope with the increasing variety management challenges could be achieved by exploiting the data records of existing or old products, along with appropriate Knowledge Discovery (KD) models, in order to extract the implicit knowledge in such data and use it to speed-up the development of new products.

Thus, the engineering problem addressed in this research is the lack of best use (e.g. efficient or complete) of the design and manufacturing data accumulated in industrial databases (e.g. Enterprise Resource Planning (ERP) and Product Life Cycle Management (PLM) systems) to support rapid development of future products. Accordingly, four related applications have been addressed through a package of KD models: product design retrieval, product family formation, assembly sequence planning and manufacturing system synthesis.

**2. Limitations of previous solutions**

*Product design Retrieval*: Group Technology (GT) classification and coding systems, and most other conventional design retrieval methods, are mainly concerned with retrieval of individual parts based on similarity of geometric features, and not the assembly level. Even existing methods that address the assembly level based on BOM trees matching, the way by which they encode and match products is computationally intense and currently could lead in many cases to misleading results.

*Product family formation*: None of the existing product family formation methods takes into consideration similarity in products structure which is important for assembly systems. It is not uncommon to find products which share many common components but with considerably different assembly structure For example, a group of control valves may share the same component (e.g. plug), but at different position and quantity for each variant.

*Assembly sequence planning*: Generating an assembly sequence without making use of previous data is an exhaustive and a time-consuming activity. As for existing retrieval-based assembly sequencing, existing methods are not able to retrieve assembly sequences for products containing groups of components that did not exist together in any of the considered individual variants data.

*Manufacturing system synthesis*: Existing association rule discovery methods used for manufacturing system are mainly based on statistical-based algorithms. These methods are quite sensitive to the used statistical thresholds; they generate huge number of redundant rules that require a lot of filtering to come up with valuable enough rules.

**3. Proposed solution**

*Product design Retrieval*: A more computationally-efficient BOM trees matching method has been developed based on well-studied and mature tree reconciliation algorithms found and used in Biological Sciences for decades. The proposed method applies a different matching approach that avoids the limitation of existing methods which leads in some situations to misleading matching results.

*Product family formation*: A product family formation method has been developed, based on a novel Bill of Material (BOM) trees similarity measure, which for the first time addresses both components commonality and hierarchical structure of products.

*Assembly sequence planning*: a new approach has been developed for assembly sequence retrieval inspired by the concept of consensus trees found in phylogenetics and evolutionary studies. It is able to retrieve assembly sequences for products containing groups of components that did not exist together in any of the considered individual variants.

*Manufacturing system synthesis*: A novel practical Integer Programming model has been developed that directly discovers robust association rules between product features and manufacturing system capabilities to be then used for manufacturing system synthesis. The proposed model is inspired by the classical set covering problem.

## 4. Proposed solution compared to previous solutions - benefits and drawbacks

Detailed comparisons between the proposed solutions and previous ones were carried out for each problem. The developed BOM trees matching method, compared to available state-of-the-art algorithms, is less computationally complex and provides more reliable results. The developed assembly sequence retrieval model extracts assembly sequencing relationships that could not be extracted by any of the existing methods. The developed model for manufacturing system synthesis based on association rule discovery is easier to implement and provides more ready-to-use results compared to existing methods. More assembly-oriented products grouping outcome is obtained using the proposed method compared with that obtained based on typical component commonality measures.

It should be mentioned that the quality of the proposed solutions is always subject to the quality of data accumulated over years based on past designers' and manufacturers' experience. Hence, planning for future products based on extracted knowledge from these data does not necessarily guaranty the best outcome. For instance, the quality of an assembly sequence plan for a new product variant, that has been developed based on assembly sequence data for similar old variants, will be limited by the quality (i.e. optimality) of that data. It should also be mentioned that the developed methods are limited to products/systems that fall within the scope of available data; human intervention is required for new products/systems that involve new features (e.g. new technology or new material) that is not part of existing database. Continuous updating of available data should improve the quality and widen the scope of extracted knowledge. For instance, continuously updating manufacturing records with manufacturing data instances involving up-to-date technologies, as well as removing manufacturing data instances involving obsolete technologies, should improve the quality and widen the scope of discovered association rules for manufacturing system synthesis.

## 6.3 Contribution

The following lines summarize contribution done in this research in each of the four applications addressed in this research.

1) Product Design Retrieval

Well-developed tree reconciliation techniques were proposed in Chapter 2 as the basis for a new method for matching Bill of Materials (BOM) trees for the purpose of retrieving similar legacy product designs. Unlike Group Technology (GT) classification and coding systems, and most other conventional design retrieval methods, which are mainly concerned with retrieval of individual parts based on similarity of geometric features, BOM matching addresses the assembly level, where similarity between whole products is sought based on their content (product components) and assembly structure.

The results of the proposed method were also compared with those found in the relevant most recent publications. This novel and straightforward BOM matching method, compared to existing methods, is less computationally complex as shown in the comparison section. The developed design retrieval method should result in considerable savings in product development time and effort.

2) Product Family Formation

In Chapter 3, as a further application for BOM trees matching, a new method for product family formation was introduced. A new BOM trees matching measure, RF-BOM$_{norm}$, which overcomes some limitations of the tree reconciliation based measure, was developed. An Integer Programming (IP) model was formulated to mathematically describe the proposed BOM trees matching measure as well as to calculate it for any given pair of BOM trees. The model is of polynomial time complexity; hence, dealing with instances involving tens of products variants and hundreds of components would be feasible.

The grouping results obtained by the proposed product family formation method were different from what could be obtained by only considering plain components commonality. From an assembly point of view, grouping based on RF-BOM$_{norm}$ results in more suitable family formation results.

3) Assembly Sequence Planning

Inspired by the concept of consensus trees found in biology and phylogenetics, a new approach for knowledge-based assembly sequencing was developed in Chapter 4. Given assembly sequence trees for a set of existing variants of a certain product family, a master (consensus) assembly sequence tree is derived. Semi-final assembly sequence trees can be extracted for new variants that fall within, or significantly overlap with, the scope/boundary of the studied family of products. Accordingly, it is then possible to generate assembly sequences for products containing groups of components that did not exist together in any of the considered individual variants. This is an advantage of generating the master assembly sequence tree which cannot be realized by retrieving the closest individual assembly sequence.

A novel Mixed-Integer Programming (MIP) model has been developed for generating the master assembly sequence tree. The model is NP-hard (non-deterministic polynomial-hard) due to its strong combinatorial and binary nature. An alternative GA-based method, with a custom designed crossover and mutation operators, was developed for potential cases that could not be handled by the MIP model in a reasonable time.

4) Manufacturing System Synthesis

An Integer Programming model is introduced in Chapter 5, for the first time, to discover association rules between product design domain and manufacturing/assembly system domain to be then used for manufacturing/assembly system synthesis. The proposed model is inspired by the classical set covering problem. The model identifies all association rules that are enough to explain any given set of manufacturing data instances.

The approach used in this model is totally different from conventional association rule discovery algorithms that look for all association rules satisfying some predefined statistical support thresholds. These methods, in addition of being quite sensitive to the used statistical thresholds, they generate huge number of redundant rules that require a lot of filtering to come up with valuable enough rules. Compared to the non-statistical co-evolution model of AlGeddawy and ElMaraghy, the developed model is more straightforward and easier to implement.

## 6.4 Future Work

Future work would address the following points:

- Considering more aspects in product design retrieval such as material of various components. This data are not usually part of the BOM and would be found in additional data files. Adding more similarity considerations should improve matching outcomes.

- Extending the knowledge-based assembly sequencing method in order to allow automatic identification of conflicting components (i.e. having multiple different assembly sequences for the same combination of components).
- Finding the best set of association rules, in cases where multiple solutions exist. Further research is still needed to investigate the best approach to realize that.
- Considering additional similarity aspects for product family formation such as demand of various product variants. This should results in more homogeneous quantities of variants to be produced within each product family which is an important aspect to avoid under utilization of assembly system resources.
- Considering the production volumes in the developed models for product family formation and knowledge-based assembly sequencing.
- Quantifying cost saving realized by using the developed knowledge-based models compared to traditional (e.g. generative) methods.

# REFERENCES

Abdi, M. R. (2012). "Product family formation and selection for reconfigurability using analytical network process." International Journal of Production Research **50**(17): 4908-4921.

Abdi, M. R. and A. W. Labib (2004). "Grouping and selecting products: the design key of reconfigurable manufacturing systems (RMSs)." International Journal of Production Research **42**(3): 521-546.

Adams, E. N. (1972). "Consensus techniques and the comparison of taxonomic trees." Systematic Zoology **21**(4): 390-397.

Agard, B. and A. Kusiak (2004). "Data-mining-based methodology for the design of product families." International Journal of Production Research **42**(15): 2955-2969.

Agrawal, R., T. Imielinski and A. Swami (1993). Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, Publ by ACM.

Agrawal, R. and R. Srikant (1994). Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, Morgan Kaufmann Publ Inc.

AlGeddawy, T. and H. ElMaraghy (2011). "Manufacturing systems synthesis using knowledge discovery." CIRP Annals - Manufacturing Technology **60**: 437-440.

AlGeddawy, T. and H. ElMaraghy (2012). "A co-evolution model for prediction and synthesis of new products and manufacturing systems." Journal of Mechanical Design, Transactions of the ASME **134**(5): 051008-051001 - 051008-051012.

Alizon, F., S. B. Shooter and T. W. Simpson (2009). "Assessing and improving commonality and diversity within a product family." Research in Engineering Design **20**(4): 241-253.

Altuntas, S., T. Dereli and H. Selim (2013). "Fuzzy weighted association rule based solution approaches to facility layout problem in cellular manufacturing system." International Journal of Industrial and Systems Engineering **15**(3): 253-271.

Asano, T., J. Jansson, K. Sadakane, R. Uehara and G. Valiente (2010). Faster computation of the Robinson-Foulds distance between phylogenetic networks. Combinatorial Pattern Matching, Springer.

Azab, A., S. Samy and H. ElMaraghy (2008). Modeling and Optimization in Assembly Planning. 2nd CIRP Conference on Assembly Technologies and Systems (CATS). Toronto, Canada.

Azadnia, A. H., S. Taheri, P. Ghadimi, M. Z. Mat Saman and K. Y. Wong (2013). "Order Batching in Warehouses by Minimizing Total Tardiness: A Hybrid Approach of Weighted Association Rule Mining and Genetic Algorithms." The Scientific World Journal **2013**.

Bai, J., S. Gao, W. Tang, Y. Liu and S. Guo (2010). "Design reuse oriented partial retrieval of CAD models." Computer-Aided Design **42**(12): 1069-1084.

Barton, J. and D. Love (2005). "Retrieving designs from a sketch using an automated GT coding and classification system." Production Planning and Control **16**(8): 763-773.

Beasley, J. E. and P. C. Chu (1996). "A genetic algorithm for the set covering problem." European Journal of Operational Research **94**(2): 392-404.

Berglund-Sonnhammer, A. C., P. Steffansson, M. J. Betts and D. A. Liberles (2006). "Optimal gene trees from sequences and species trees using a soft interpretation of parsimony." Journal of molecular evolution **63**(2): 240-250.

Bisschop, J. (2005). AIMMS-Optimization modeling. Haarlem, The Netherlands, Paragon Decision Technology.

Blecker, T. and N. Abdelkafi (2007). "The development of a component commonality metric for mass customization." Engineering Management, IEEE Transactions on **54**(1): 70-85.

Bonizzoni, P., G. D. Vedova and R. Dondi (2005). "Reconciling a gene tree to a species tree under the duplication cost model." Theoretical Computer Science **347**(1): 36-53.

Bourgon, R., M. Delorenzi, T. Sargeant, A. N. Hodder, B. S. Crabb and T. P. Speed (2004). "The serine repeat antigen (SERA) gene family phylogeny in Plasmodium: the impact of GC content and reconciliation of gene and species trees." Molecular biology and evolution **21**(11): 2161-2171.

Bryant, D. (2003). "A classification of consensus methods for phylogenetics." DIMACS series in discrete mathematics and theoretical computer science **61**: 163-184.

Buddhakulsomsiri, J., Y. Siradeghyan, A. Zakarian and X. Li (2006). "Association rule-generation algorithm for mining automotive warranty data." International Journal of Production Research **44**(14): 2749-2770.

Cardona, G., F. Rossello and G. Valiente (2009). "Comparison of tree-child phylogenetic networks." IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **6**(4): 552-569.

Ceglar, A. and J. F. Roddick (2006). "Association mining." ACM Computing Surveys **38**(2): 42 pp.

Chang, S.-H., W.-L. Lee and R.-K. Li (1997). "Manufacturing bill-of-material planning." Production Planning and Control **8**(5): 437-450.

Chawathe, S. S. and H. Garcia-Molina (1997). Meaningful change detection in structured data. SIGMOD 1997. ACM SIGMOD International Conference on Management of Data, USA, ACM.

Chen, K., D. Durand and M. Farach-Colton (2000). "NOTUNG: a program for dating gene duplications and optimizing gene family trees." Journal of Computational Biology **7**(3-4): 429-447.

Chen, W.-C., S.-S. Tseng and C.-Y. Wang (2004). A novel manufacturing defect detection method using data mining approach. 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004, Berlin, Germany, Springer-Verlag.

Chieh-Yuan, T. and C. A. Chang (2003). "Fuzzy neural networks for intelligent design retrieval using associative manufacturing features." Journal of Intelligent Manufacturing **14**(2): 183-195.

Choudhary, A. K., J. A. Harding and M. K. Tiwari (2009). "Data mining in manufacturing: a review based on the kind of knowledge." Journal of Intelligent Manufacturing **20**(5): 501-521.

Da Cunha, C., B. Agard and A. Kusiak (2006). "Data mining for improvement of product quality." International Journal of Production Research **44**(18-19): 4027-4041.

Danicic, D., D. Durand, A. Goldman, M. Stolzer and V. Benjamin (2008). Notung 2.6: A Manual, Durand Lab.

De Fazio, T. L. and D. E. Whitney (1987). "Simplified Generation of All Mechanical Assembly Sequences." IEEE journal of robotics and automation **RA-3**(6): 640-658.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE Transactions on Evolutionary Computation **6**(2): 182-197.

Dini, G., F. Failli, B. Lazzerini and F. Marcelloni (1999). "Generation of optimized assembly sequences using genetic algorithms." CIRP Annals - Manufacturing Technology **48**(1): 17-20.

Dong, J., D. Fernandez-Baca, F. R. McMorris and R. C. Powers (2010). "Majority-rule (+) consensus trees." Mathematical Biosciences **228**(1): 10-15.

Dong, T., R. Tong, L. Zhang and J. Dong (2005). "A collaborative approach to assembly sequence planning." Advanced Engineering Informatics **19**(2): 155-168.

Durand, D., B. V. Halldórsson and B. Vernot (2006). "A hybrid micro-macroevolutionary approach to gene tree reconstruction." Journal of Computational Biology **13**(2): 320-335.

Eguia, I., S. Lozano, J. Racero and F. Guerrero (2011). "A methodological approach for designing and sequencing product families in Reconfigurable Disassembly Systems." Journal of Industrial Engineering and Management **4**(3): 418-435.

ElMaraghy, H., Ed. (2009). Changeable and Reconfigurable Manufacturing Systems. Springer Series in Advanced Manufacturing, Springer.

ElMaraghy, H., S. Samy and V. Espinoza (2010). A classification code for assembly systems. 3rd CIRP Conference on Assembly Technologies and Systems, CATS2010, Trondheim, Norway, Marcel Dekker.

ElMaraghy, H., G. Schuh, W. ElMaraghy, F. Piller, P. Schönsleben, M. Tseng and A. Bernard (2013). "Product variety management." CIRP Annals - Manufacturing Technology **62**(2): 629-652.

ElMaraghy, H. A. (1993). "Evolution and Future Perspectives of CAPP." CIRP Annals - Manufacturing Technology **42**(2): 739-751.

ElMaraghy, H. A. and T. AlGeddawy (2012). "Co-evolution of products and manufacturing capabilities and application in auto-parts assembly." Flexible Services and Manufacturing Journal **24**(2): 142-170.

ElMaraghy, H. A. and W. H. ElMaraghy (2014). Variety, Complexity and Value Creation. Enabling Manufacturing Competitiveness and Economic Sustainability. M. F. Zaeh, Springer**:** 1-7.

ElMaraghy, H. A. and L. Knoll (1991). "Design and automatic assembly sequence generation of a d.c. motor." International Journal of Vehicle Design **12**(5-6): 672-683.

ElMaraghy, H. A., O. Kuzgunkaya and R. J. Urbanic (2005). "Manufacturing systems configuration complexity." CIRP Annals - Manufacturing Technology **54**(1): 445-450.

ElMaraghy, H. A. and L. Laperriere (1992). "Modelling and sequence generation for robotized mechanical assembly." Robotics and Autonomous Systems **9**(3): 137-147.

ElMaraghy, H. A. and J. M. Rondeau (1992). "Automatic robot program synthesis for assembly." Robotica **10**: 113-123.

ElMaraghy, W. (2009). "Knowledge Management in collaborative engineering." International Journal of Collaborative Engineering **1**(1): 114-124.

Fayyad, U., G. Piatetsky-Shapiro and P. Smyth (1996). "From data mining to knowledge discovery in databases." AI magazine **17**(3): 37-54.

Fitch, W. M. (1971). "Toward defining the course of evolution: minimum change for a specific tree topology." Systematic Biology **20**(4): 406-416.

Gabaldon, T. (2006). "Computational approaches for the prediction of protein function in the mitochondrion." American Journal of Physiology-Cell Physiology **291**(6): C1121-C1128.

Galan, R., J. Racero, I. Eguia and J. M. Garcia (2007). "A systematic approach for product families formation in Reconfigurable Manufacturing Systems." Robotics and Computer-Integrated Manufacturing **23**(5): 489-502.

Godinho Filho, M., C. F. Barco and R. F. Tavares Neto (2014). "Using Genetic Algorithms to solve scheduling problems on flexible manufacturing systems (FMS): A literature survey, classification and analysis." Flexible Services and Manufacturing Journal **26**(3): 408-431.

Gongzhuang, P., M. Huachao, L. Minghui and Z. Heming (2014). Multi-level knowledge representation and retrieval of complex product design based on BOM. IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Piscataway, NJ, USA, IEEE.

Gonzalez, B. and B. Adenso-Diaz (2005). "A bill of materials-based approach for end-of-life decision making in design for the environment." International Journal of Production Research **43**(10): 2071-2099.

Gouwanda, D. and S. Ponnambalam (2008). "Evolutionary search techniques to solve set covering problems." World Academy of Science, Engineering and Technology **39**(4): 20-25.

Goyal, K. K., P. Jain and M. Jain (2013). "A comprehensive approach to operation sequence similarity based part family formation in the reconfigurable manufacturing system." International Journal of Production Research **51**(6): 1762-1776.

Gupta, A., P. Jain and D. Kumar (2013). "A novel approach for part family formation using K-means algorithm." Advances in Manufacturing **1**(3): 241-250.

Gupta, S. and V. Krishnan (1998). "Product family-based assembly sequence design methodology." IIE transactions **30**(10): 933-945.

Holland, J. H. (1992). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence, MIT press.

Homem de Mello, L. S. and A. C. Sanderson (1989). A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE International Conference on Robotics and Automation, Washington, DC, USA, IEEE Comput. Soc. Press.

Hong, D. S. and H. S. Cho (1997). "Generation of robotic assembly sequences with consideration of line balancing using simulated annealing." Robotica **15**: 663-673.

Huang, G. Q., Y. Zhang, X. Chen and S. T. Newman (2008). "RFID-enabled real-time wireless manufacturing for adaptive assembly planning and control." Journal of Intelligent Manufacturing **19**(6): 701-713.

Hui, Z. (2011). "The application of knowledge discovery in triggering design inspiration." Applied Mechanics and Materials **88-89**: 170-174.

Jain, A. K., M. N. Murty and P. J. Flynn (1999). "Data clustering: a review." ACM computing surveys (CSUR) **31**(3): 264-323.

Jiang, T., W. Lusheng and Z. Kaizhong (1995). "Alignment of trees-an alternative to tree edit." Theoretical Computer Science **143**(1): 137-148.

Jiao, J., M. Tseng, Q. Ma and Z. Yi (2000). "Generic bill-of-materials-and-operations for high-variety production management." Concurrent Engineering: Research and Applications **8**(4): 297-321.

Jiao, J., L. Zhang, Y. Zhang and S. Pokharel (2008). "Association rule mining for product and process variety mapping." International Journal of Computer Integrated Manufacturing **21**(1): 111-124.

Jiao, J. and Y. Zhang (2005). "Product portfolio identification based on association rule mining." CAD Computer Aided Design **37**(2): 149-172.

Jimenez, P. (2013). "Survey on assembly sequencing: A combinatorial and geometrical perspective." Journal of Intelligent Manufacturing **24**(2): 235-250.

Jones, R. E., R. H. Wilson and T. L. Calton (1998). "On constraints in assembly planning." IEEE Transactions on Robotics and Automation **14**(6): 849-863.

Kneppelt, L. R. (1984). "Product structuring considerations for master production scheduling." Production and Inventory Management **25**(1): 83-99.

Kunpeng, Z., W. Yoke San, L. Han Tong and L. Wen Feng (2012). "3D CAD model retrieval with perturbed Laplacian spectra." Computers in industry **63**(1): 1-11.

Kusiak, A. (2002). "A data mining approach for generation of control signatures." Transactions of the ASME. Journal of Manufacturing Science and Engineering **124**(4): 923-926.

Kusiak, A. (2006). "Data mining in manufacturing: a review." Journal of Manufacturing Science and Engineering **128**(4): 969-976.

Lai, H.-Y. and C.-T. Huang (2003). "Integrated assembly plan generation system for grouped product families." International Journal of Production Research **41**(17): 4041-4061.

Lakhal, L. and G. Stumme (2005). Efficient mining of association rules based on formal concept analysis. Formal Concept Analysis. Foundations and Applications. Berlin, Germany, Springer-Verlag**:** 180-195.

Laperriere, L. and H. A. ElMaraghy (1994). "Assembly sequences planning for simultaneous engineering applications." International Journal of Advanced Manufacturing Technology **9**(4): 231-244.

Lee-Post, A. (2000). "Part family identification using a simple genetic algorithm." International Journal of Production Research **38**(4): 793-810.

Lim, C., Y. Lee and J. Choi (2006). "Database construction and data retrieval for optimal casting design." International Journal of Cast Metals Research **19**(4): 259-264.

Mahamaneerat, W. K., S. Chi-Ren, H. Shih-Chun and C. A. Chang (2007). "Domain-concept association rules mining for large-scale and complex cellular manufacturing tasks." Journal of Manufacturing Technology Management **18**(7): 787-806.

Maki, H. and Y. Teranishi (2001). Development of automated data mining system for quality control in manufacturing. Proceedings of DaWaKOl - 3rd International Conference on Data Warehousing and Knowledge Discovery, Berlin, Germany, Springer-Verlag.

Marian, R. M., L. H. S. Luong and K. Abhary (2002). "Assembly sequence planning and optimisation using genetic algorithms.I. Automatic generation of feasible assembly sequences." Applied Soft Computing **2**(2): 223-253.

Martinez, M. T., J. Favrel and P. Ghodous (2000). "Product family manufacturing plan generation and classification." Concurrent Engineering Research and Applications **8**(1): 12-23.

Maskell, B. H. (1988). "Bills of Material for Engineering Production and Costing." Production and Inventory Management Review **8**(5): 38-44.

Mather, H. (1987). Bills of Materials. Homewood, IL, Dow Jones-Irwin.

McAuley, J. (1972). "Machine grouping for efficient production." Production Engineer **51**(2): 53-57.

Miguel, P. A. C. (2005). "Modularity in product development: a literature review towards a research agenda." Product: Management & Development **3**(2): 165-174.

Miller, J. M. and R. L. Hoffman (1989). Automatic assembly planning with fasteners. IEEE International Conference on Robotics and Automation, Scottsdale, AZ, USA, Publ by IEEE.

Nilsson, N. J. (1980). Principles of artificial intelligence, Springer-Verlag.

Opitz, H. (1970). A classification system to describe workpieces. Oxford, Pergamon Press.

Orlicky, J. A. (1971). Material Requirements Planning. New York, McGraw-Hill.

Orlicky, J. A., G. W. Plossl and O. W. Wight (1972). "Structuring the bill of material for MRP." Production and Inventory Management **13**: 19-42.

Page, R. D. (2003). Tangled trees: Phylogeny, cospeciation, and coevolution, University of Chicago Press.

Page, R. D. M. (1994). "Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas." Systematic Biology **43**(1): 58-77.

Page, R. D. M. and M. A. Charleston (1997). "From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem." Molecular phylogenetics and evolution **7**(2): 231-240.

Pan, C., S. S. Smith and G. C. Smith (2006). "Automatic assembly sequence planning from STEP CAD files." International Journal of Computer Integrated Manufacturing **19**(8): 775-783.

Patel, P. C. and J. Jayaram (2014). "The antecedents and consequences of product variety in new ventures: An empirical study." Journal of Operations Management **32**(1-2): 34-50.

Pattanaik, L. N. and V. Kumar (2011). "Product family formation for reconfigurable manufacturing using A bi-criterion evolutionary algorithm." International Journal of Industrial Engineering : Theory Applications and Practice **18**(9): 493-505.

Pattengale, N. D., E. J. Gottlieb and B. M. Moret (2007). "Efficiently computing the Robinson-Foulds metric." Journal of Computational Biology **14**(6): 724-735.

Piatetsky-Shapiro, G. (1991). "Knowledge discovery in real databases: A report on the IJCAI-89 Workshop." AI magazine **11**(5): 68-70.

Rajagopalan, S. and N. Xia (2012). "Product variety, pricing and differentiation in a supply chain." European Journal of Operational Research **217**(1): 84-93.

Rakesh, K., P. Jain and N. Mehta (2010). "A framework for simultaneous recognition of part families and operation groups for driving a reconfigurable manufacturing system." Advances in Production Engineering & Management Journal **5**(1): 45-58.

Robinson, D. F. and L. R. Foulds (1981). "Comparison of phylogenetic trees." Mathematical Biosciences **53**(1-2): 131-147.

Romanowski, C., R. Nagi and M. Sudit (2006). "Data mining in an engineering design environment: OR applications from graph matching." Computers & operations research **33**(11): 3150-3160.

Romanowski, C. J. and R. Nagi (2004). "A data mining approach to forming generic bills of materials in support of variant design activities." Journal of Computing and Information Science in Engineering **4**(4): 316-328.

Romanowski, C. J. and R. Nagi (2005). "On comparing bills of materials: a similarity/distance measure for unordered trees." IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans) **35**(2): 249-260.

Ruo, H. and X. Z. Fu (2011). Classification of knowledge discovery methods. 2011 International Conference on Mechanical Engineering, Industry and Manufacturing Engineering, MEIME2011, Beijing, China, Trans Tech Publications.

Saaty, T. L. (1986). "Axiomatic foundation of the analytic hierarchy process." Management science **32**(7): 841-855.

Saaty, T. L. (1996). Decision making with dependence and feedback: The analytic network process. Pittsburgh, Pennsylvania, RWS Publications.

Sadoyan, H., A. Zakarian and P. Mohanty (2006). "Data mining algorithm for manufacturing process control." International Journal of Advanced Manufacturing Technology **28**(3-4): 342-350.

Samy, S. N. and H. ElMaraghy (2012). "A model for measuring complexity of automated and hybrid assembly systems." International Journal of Advanced Manufacturing Technology **62**(5-8): 813-833.

Searls, D. B. (2003). "Pharmacophylogenomics: genes, evolution and drug targets." Nature Reviews Drug Discovery **2**(8): 613-623.

Seifoddini, H. and P. M. Wolfe (1986). "Application of the similarity coefficient method in group technology." IIE transactions **18**(3): 271-277.

Shao, X. Y., Z. H. Wang, P. G. Li and C. X. J. Feng (2006). "Integrating data mining and rough set for customer group-based discovery of product configuration rules." International Journal of Production Research **44**(14): 2789-2811.

Shasha, D., J. T.-L. Wang, K. Zhang and F. Y. Shih (1994). "Exact and approximate algorithms for unordered tree matching." IEEE Transactions on Systems, Man and Cybernetics **24**(4): 668-678.

Shih, H. M. (2011). "Product structure (BOM)-based product similarity measures using orthogonal procrustes approach." Computers & Industrial Engineering **61**(3): 608-628.

Sinanoglu, C. and H. R. Borklu (2005). "An assembly sequence-planning system for mechanical parts using neural network." Assembly Automation **25**(1): 38-52.

Song, Z. and A. Kusiak (2009). "Optimising product configurations with a data-mining approach." International Journal of Production Research **47**(7): 1733-1751.

Steva, E. D., E. N. Rice, T. J. Marion, T. W. Simpson and R. B. Stone (2006). Two methodologies for identifying product platform elements within an existing set of products. ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.

Stonebraker, P. W. (1996). "Restructuring the bill of material for productivity: a strategic evaluation of product configuration." International Journal of Production Economics **45**(1-3): 251-260.

Stumme, G., R. Taouil, Y. Bastide, N. Pasquier and L. Lakhal (2002). "Computing iceberg concept lattices with TITANIC." Data & Knowledge Engineering **42**: 189-222.

Takenaka, T., H. Koshiba, Y. Motomura and K. Ueda (2013). "Product/service variety strategy considering mixed distribution of human lifestyles." CIRP Annals - Manufacturing Technology **62**(1): 463-466.

Thevenot, H. J. and T. W. Simpson (2007). "A comprehensive metric for evaluating component commonality in a product family." Journal of Engineering Design **18**(6): 577-598.

Ulrich, K. T. and S. D. Eppinger (2011). Product design and development. New York, NY, McGraw-Hill/Irwin.

Van Veen, E. A. and J. C. Wortmann (1987). "Generic bills of materials in assemble-to-order manufacturing " International Journal of Production Research **25**(11): 1645-1658.

Vernot, B., M. Stolzer, A. Goldman and D. Durand (2008). "Reconciliation with non-binary species trees." Journal of Computational Biology **15**(8): 981-1006.

Wang, J. F., J. H. Liu and Y. F. Zhong (2005). "A novel ant colony algorithm for assembly sequence planning." International Journal of Advanced Manufacturing Technology **25**(11-12): 1137-1143.

Wang, Y. and J. H. Liu (2010). "Chaotic particle swarm optimization for assembly sequence planning." Robotics and Computer-Integrated Manufacturing **26**(2): 212-222.

Wang, Z., X. Shao, G. Zhang and H. Zhu (2005). Integration of variable precision rough set and fuzzy clustering: An application to knowledge acquisition for manufacturing process planning. 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, RSFDGrC 2005, Regina, Canada, Springer Verlag.

Whitney, D. E. (2004). Mechanical assemblies: their design, manufacture, and role in product development, Oxford University Press, USA.

Wolter, J. D. (1992). "A combinatorial analysis of enumerative data structures for assembly planning." Journal of Design and Manufacturing **2**(2): 93-104.

Yan, C. and W. Shi (2013). Association rule mining for job shop scheduling problem based on genetic algorithm. 2012 International Conference on Material Sciences and Manufacturing Technology, ICMSMT 2012, Dalian, China, Trans Tech Publications.

Zaki, M. J. (2004). "Mining non-redundant association rules." Data Mining and Knowledge Discovery **9**(3): 223-248.

# APPENDICES

## APPENDIX A

**Table A1:** Encoded product variants for the engine block example in Chapter 5

| | | Engine Variants | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **TW3-EZX** | **TW3-EZY** | **TW3-EZV** | **TW3-EZN** | **TW3-EZM** | **TW5-ABA** | **TW5-ABB** | **TW5-ABM** | **TW5-ABE** | **TW3-EZX** |
| **Products Features** | **1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **2** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **3** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **4** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **5** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **6** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **7** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | **8** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **9** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **10** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Table A2:** Encoded assembly systems for the engine block example in Chapter 5

| | | | **Engine Variants** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **TW3-EZX** | **TW3-EZY** | **TW3-EZV** | **TW3-EZN** | **TW3-EZM** | **TW5-ABA** | **TW5-ABB** | **TW5-ABM** | **TW5-ABE** |
| **Assembly System Stations** | **1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **2** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **3** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **4** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **5** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **6** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **7** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | **8** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | **9** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | **10** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | **12** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | **13** | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **14** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | **15** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | **16** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | **17** | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| | **18** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **19** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **20** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | **21** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | **22** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | **23** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | **24** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Table A3:** Obtained association matrix for the engine block example in Chapter 5

| Assembly System Stations | | Product Features | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | | | | | 2 | | 3 | | | | 4 | | | | 5 | | | | 6 | | | | 7 | | | 8 | | 9 | | | | | 10 | | | |
| | | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 22 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 23 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# VITA AUCTORIS

| | |
|---|---|
| NAME: | Mohamed Kashkoush |
| PLACE OF BIRTH: | Cairo, Egypt |
| YEAR OF BIRTH: | 1984 |
| EDUCATION: | B.Sc. in Mechanical Design and Production Engineering, Cairo University, Giza, Egypt, 2006 |
| | M.Sc. in Industrial Engineering, Cairo University, Giza, Egypt, 2010 |

PUBLICATIONS: **Journal Publications:**

**1.** Kashkoush, M. and ElMaraghy, H., 2014. Product Design Retrieval by Matching Bills of Materials. Journal of Mechanical Design, Transactions of the ASME, Vol. 136(1), pp. 011002-1 - 011002-10.

**2**. Kashkoush, M. and ElMaraghy, H., 2014. Consensus Tree Method for Generating Master Assembly Sequence. Production Engineering, Vol. 8, pp. 233–242.

**3.** Kashkoush, M. and ElMaraghy, H., 2015. Knowledge-Based Model for Constructing Master Assembly Sequence. Journal of Manufacturing Systems, Vol. 34, pp. 43-52.

**4.** Kashkoush, M. and ElMaraghy, H., 2015. An Integer Programming Model for Discovering Associations between Manufacturing System Capabilities and Product Features. Journal of Intelligent Manufacturing, DOI: 10.1007/s10845-015-1044-6.

**5.** Shalaby, M., and Kashkoush, M., 2013. A Particle Swarm Optimization Algorithm for a 2-D Irregular Strip Packing Problem. American Journal of Operations Research, Vol. 3, pp. 268-278.

**6.** Kashkoush, M., Shalaby, M., and Abdelhafiez, E., 2012. A Mixed-Integer Model for Two-Dimensional Polyominoes Strip Packing and Tiling Problems. International Journal of Operational Research, Vol. 15(4), pp. 391-405.

**7.** Kashkoush, M. and ElMaraghy, H. 2015. Product Family Formation by Matching BOM Trees. Submitted to CIRP Journal of Manufacturing Science and Technology, April 2014, Ref. no.: CIRPJ-D-14-00050.

**8.** ElMaraghy, H. and Kashkoush, M., 2015. Assembly System Synthesis Using Association Rule Discovery. Accepted for publication in The International Journal of Journal of Advanced Manufacturing Technology, April 2015, Ref. no.: JAMT-D-15-00351.

**Conference Publications:**

**1.** Kashkoush, M. and ElMaraghy, H., 2014. Product Family Formation for Reconfigurable Assembly Systems. In proceeding of the 47th CIRP Conference on Manufacturing Systems (CMS), Windor, ON, Canada, Procedia CIRP, Vol. 17, pp. 302-307.

**2.** Kashkoush, M. and ElMaraghy, H., 2013. Matching Bills of Materials Using Tree Reconciliation. In proceeding of the 46th CIRP Conference on Manufacturing Systems (CMS), Setubal, Portugal, Procedia CIRP, Vol. 7C, pp. 169-174.

**3.** Kashkoush, M. and ElMaraghy, H., 2013. Generating Master Assembly Sequence Using Consensus Trees. In proceeding of the 5th Int. Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV), Munich, Germany, pp. 261-266.

**4.** Kashkoush, M., AlGeddawy, T., and ElMaraghy, H., 2012. Discovering Design Structure Matrix for Family of Products. In proceedings of the 4th CIRP Conf. on Assembly Technologies and Systems (CATS), Ann Arbor, MI, USA, pp. 191-194.

**5.** Kashkoush, M., Shalaby, M. and Abdelhafiez, E., 2010. A Combined Pixel-Compaction Placement Algorithm for the 2-D Irregular Strip Packing Problem. In proceedings of the 8th Int. Conf. of Ain Shams Uni. on Production Engineering & Design for Development (PEDD), Cairo, Egypt, pp. 108 – 120.