

## AN ABSTRACT OF THE THESIS OF

Yu-An Li for the degree of Doctor of Philosophy in  
Industrial Engineering presented on March 19, 1999.

Title: Component To Multi-Track Feeder Assignment And Board Sequencing For  
Printed Circuit Board Assembly

Abstract approved: \_\_\_\_\_  
Sabah U. Randhawa

This research was motivated by the use of multi-track feeders in the printed circuit board (PCB) assembly. In a low volume, high mix production environment, setup time is usually considered more important than processing time. Implementation of multi-track feeders not only increases the capacity of the surface mount machines but also reduces feeder changeovers. However, improper planning could diminish these benefits.

The objective of this research is to develop a process plan to minimize the feeder setups in multi-track feeder systems. Two problems have been identified: component to multi-track feeder assignment problem and PCB sequencing problem. The assignment problem is formulated as a multi-dimension symmetric assignment problem with an integer-programming model. The objective is to maximize the total similarity of the component assignment. This optimization model is implemented for small-sized problems using a commercial solver package. Due to *NP*-complete characteristics, heuristic algorithms are developed for solving large-scale problems and industrial cases. The Hungarian algorithm, designed for asymmetric assignment problems, is used to reduce problem size in the double feeder case.

The PCB sequencing problem is solved in three stages: component and PCB grouping, intra- and inter-group PCB sequencing, and feeder setup planning. An optimal tool switch policy called *Keep Tool Needed Soonest* is adapted for planning

the multi-track feeder setup. This research also identifies the interrelationship of the assignment problem and PCB sequencing problem. An optimal component to feeder assignment will show real advantages only when working with a well-planned PCB sequence.

Data obtained from literature are used to verify the heuristic developments. The methods are also applied to industrial data for evaluation of performance of real-world problems. Experimentation is conducted with simulation data to investigate the performance of methodology for different production situations. The results show that savings of up to 85% in feeder setups can be realized with a double feeder system compared to a single feeder system, with the use of the developed methodology. The approach is also robust and efficient for different production environments.

©Copyright by Yu-An Li

March 19, 1999

All rights reserved

Component To Multi-Track Feeder Assignment And Board Sequencing  
For Printed Circuit Board Assembly

by

Yu-An Li

A THESIS

submitted to

Oregon State University

in partial fulfillment of  
the requirements for the  
degree of

Doctor of Philosophy

Completed March 19, 1999  
Commencement June 1999

Doctor of Philosophy thesis of Yu-An Li presented on March 19, 1999

APPROVED:

---

Major Professor, representing Industrial Engineering

---

Head of the Department of Industrial Engineering

Redacted for privacy


---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

---

 Yu-An Li, Author

## ACKNOWLEDGMENT

Gratitude is expressed toward my major professor, Dr. Sabah Randhawa, for his encouragement and tremendous supports. I am grateful for his friendship and assistance for those years of studies at Oregon State University.

Appreciation is also given to other members of my committee, Drs. Brian Paul, Rick Wilson, David Thomas, and Joseph Zaworski for their helps in completing the research. Also, I would like to thank Tektronix, Inc., in particular to Dr. Zhongkai Xu for providing data sets.

I would also like to thank Chi-Shih Wu and Jung-Tsu Lin for their helps through these years. I also want to thank Mrs. Julia Gray for her kindness and moral supports.

My Ph.D. studies would not be complete without the support of my parents who encourage me for education and support me in every aspect without conditions.

Finally, I would like to express my most heartfelt thanks to my wife, Hui-I Chung, for her faith in me and sacrifices and continuous support. Without her, I cannot possibly approach this achievement.

## CONTRIBUTION OF AUTHORS

Dr. Sabah Randhawa was involved in the analysis and writing of each manuscript. Dr. Zhongkai Xu was involved in development and analysis of the component to multi-track feeder assignment problem and assisted in data collection for this study.

## TABLE OF CONTENTS

	<u>Page</u>
CHAPTER 1 INTRODUCTION.....	1
1.1 Overview of PCB Assembly Process and Surface Mount Technology .....	1
1.2 PCB Assembly Systems.....	3
1.3 PCB Assembly Process Planning.....	7
1.4 Research Objectives.....	9
1.5 Research Contributions .....	10
1.6 Organization of Dissertation .....	11
CHAPTER 2 PROBLEM STATEMENT AND ASSUMPTIONS .....	12
2.1 Multi-Track Feeder Configuration.....	12
2.2 Component-Multi-track Feeder Assignment Problem.....	13
2.2.1 Single Feeder and Setup Reduction.....	14
2.2.2 Double Feeder and Setup Reduction .....	15
2.2.3 Problem Assumptions.....	17
2.3 Multi-Track Feeder and Board Sequencing Problems.....	19
2.4 Overall Problem Statement and Assumptions .....	20
2.5 Current Industry Application .....	21
CHAPTER 3 BACKGROUND AND LITERATURE REVIEW .....	23
3.1 Trade-off Between Process Time and Setup Time .....	23
3.2 Group Technology .....	26
3.3 Assignment Problem.....	30
3.3.1 Combinatorial Optimization.....	32
3.3.2 NP-Complete Class Problem.....	33
3.3.3 Exact and Heuristic Algorithm.....	34
3.4 PCB Sequencing Problem.....	35



## TABLE OF CONTENTS (Continued)

	<u>Page</u>
3.5 Interrelated Problems .....	38
 CHAPTER 4 COMPONENT TO MULTI-TRACK FEEDER ASSIGNMENT PROBLEM .....	
4.1 Methodology .....	41
4.1.1 Example Illustration .....	43
4.1.2 Methodology Description .....	45
4.2 Computation Modeling .....	47
4.2.1 Multi-track Feeder Assignment Model – Generalized IP Models .....	47
4.2.2 Multi-track Feeder Assignment Model – Application Models .....	48
4.2.3 Implementation of the IP Model .....	50
4.2.4 Discussion .....	52
4.3 Heuristic Development .....	53
4.3.1 Min-Max (MMX) Heuristic – A Greedy Search Heuristic Algorithm .....	54
4.3.2 SWAP – A Neighborhood Search Heuristic Algorithm .....	55
4.3.3 Integrated Heuristic with Problem Size Reduction (Double feeder case) .....	56
4.4 Evaluation Study .....	62
4.4.1 Description of Test Problems .....	62
4.4.2 Computational Results and Analysis .....	64
4.5 Chapter Summary and Conclusions .....	72
 CHAPTER 5 PCB SEQUENCING PROBLEM WITH MULTI-TRACK FEEDERS .....	
5.1 Methodology .....	74
5.2 Component and PCB Grouping .....	78
5.2.1 Example Illustration .....	82
5.2.2 Discussion .....	87
5.3 Intra and Inter-Group Sequencing Method (IIGS) .....	88

## TABLE OF CONTENTS (Continued)

	<u>Page</u>
5.3.1 Sequencing Procedure .....	89
5.3.2 Example Illustration .....	89
5.3.3 Discussion.....	92
5.4 Optimal Component Switch Plan - KTNS.....	92
5.4.1 KTNS Procedure .....	92
5.4.2 Modified KTNS for Multi-track feeder Configuration .....	96
5.5 Application illustration .....	98
5.5.1 Single Feeder Setup Plan.....	98
5.5.2 Double-Feeder Setup Plan.....	100
5.6 Evaluation of the Methodology .....	102
5.6.1 Evaluation on Example Data.....	104
5.6.2 Evaluation Using Industry Data .....	106
5.7 Chapter Summary and Conclusions.....	108
CHAPTER 6 EXPERIMENTATION AND ANALYSIS.....	109
6.1 Industry Data Analysis.....	109
6.2 Description of the Simulation Data.....	111
6.2.1 Random Number Generator for Incidence Matrix .....	111
6.2.2 Experiment Data Configurations.....	112
6.3 Experiment Design and Results .....	115
6.3.1 Experiment Results.....	116
6.3.2 Analysis of Effects on Feeder Setup .....	117
6.3.3 Analysis of Effects on Solution Times .....	120
6.4 Evaluation of Algorithm Combinations.....	123
6.5 Chapter Summary and Conclusions.....	127
CHAPTER 7 SUMMARY AND CONCLUSIONS.....	129
7.1 Summary of the Dissertation .....	129

## TABLE OF CONTENTS (Continued)

	<u>Page</u>
7.2 Conclusions.....	130
7.3 Suggestions for Future Research .....	132
7.3.1 Integration with Component Placement Sequence Problem .....	133
7.3.2 Integration with Feeder Allocation Problem .....	133
7.3.3 Application in Limited Resource Case.....	134
7.3.4 Integration with Workload Balancing Problem.....	135
REFERENCES .....	136
APPENDICES .....	143
APPENDIX A.....	144
APPENDIX B .....	158
APPENDIX C .....	173

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1.1 PCB assembly procedures .....	4
1.2 Pick-and-place machine with (a) rotary turret design (b) gantry design .....	5
1.3 Relationships of process information and planning .....	8
2.1 (a) A single feeder, and (b) A double feeder [13] .....	13
3.1 Optimization strategy and the importance of process versus setup times. ....	24
4.1 Integrated Heuristic Flow and Setup Options .....	61
4.2 The number of (a) best solutions, and (b) optimal solutions found by each heuristics. ....	66
4.3 Dependency of computation time on data size, distribution, and heuristics.....	67
4.4 The relationships of data sets and fitted equation parameters $\alpha$ and $\beta$ in each heuristic. ....	69
4.5 The percentage of a) symmetric similarity values, and b) number of symmetric assignments, solved by the first reduction procedure.....	71
5.1 Methodology flowchart .....	76
5.2 Component grouping procedures .....	79
5.3 PCB grouping procedures.....	80
5.4 IIGS PCB sequencing procedures .....	90
5.5 KTNS procedures for single feeder configuration .....	94
5.6 Modified KTNS procedures for double-feeder systems.....	97
5.7 Evaluation results .....	104
6.1 The component usage and board requirement histograms for Data C .....	110
6.2 The USG and REQ histograms for industry simulated data EMM. ....	114
6.3 Example of high deviation USG and low deviation REQ (EHL). ....	114

## LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Page</u>
6.4 Visual comparisons of industry data and simulation data .....	115
6.5 Data transformation for Feeder setup .....	117
6.6 a) Cube plot, b) main effect plot, and c) interaction plot of Assignment, USG stdev., and Sequence for log(Feeder setup).....	119
6.7 a) main effect plot, b) interaction plot of all factors for CPU times.....	121
6.8 Cube plot of interaction a) ABC, and b) ACD for CPU times .....	122
6.9 Evaluation results of feeder setup for data set 2.....	124
6.10 Evaluation results of CPU times for data set 2.....	125
6.11 Evaluation results for industry data set C.....	127

## LIST OF TABLES

<u>Table</u>	<u>Page</u>
1.1 Comparison of general high-throughput and high-flexibility machines .....	6
2.1 Example of a PCB incidence matrix .....	14
2.2 Example of a single feeder setup .....	15
2.3 Example of double feeder setup with different assignments .....	16
2.4 Example of double feeder setup with different PCB sequences .....	19
3.1 Comparison of commonly used heuristics .....	35
4.1 Example of PCB incidence matrix and double feeder assignments reproduced from Table 2.1 and 2.3 .....	42
4.2 Similarity matrix for double-feeder example .....	43
4.3 Similarity measurements for double-feeder example .....	44
4.4 Comparison of general and simplified assignment models .....	50
4.5 Data Set 1 from OR-Library .....	62
4.6 Data Set 2 from the random number generator .....	63
4.7 Data Set 3 from industry .....	64
4.8 Computational results for the Data set 1 .....	65
4.9 The fitted functions for Data set 2 .....	68
4.10 Comparisons of actual and predicted computation time for industry data ...	70
5.1 Tabular method for the first 10 iterations .....	83
5.2 Calculation illustrating for component grouping .....	85
5.3 Component and PCB groups .....	87
5.4 Component and PCB groups (duplicated from Appendix B.6, Set 1) .....	89
5.5 Illustration of KTNS procedure .....	99
5.6 Illustration of modified KTNS procedure on double feeder case .....	101

## LIST OF TABLES (Continued)

<u>Table</u>	<u>Page</u>
5.7 Main and interaction effects of evaluated methods .....	103
5.8 Interpretation of comparisons from evaluation .....	103
5.9 Results from example data .....	105
5.10 Results from example data (section 5.6.1) and industry data sets .....	107
6.1 Industry data set descriptions .....	110
6.2 Experimentation data set descriptions .....	113
6.3 Summary of $2^4$ factorial design .....	116
6.4 Final fitted model for log(Feeder setup).....	118
6.5 Backward selection stepwise regression for CPU times. ....	120
6.6 Similarity values of each assignment .....	123

## LIST OF APPENDICES

<u>Appendix</u>	<u>Page</u>
A.1 AMPL Model for Double Feeder Assignment Problem.....	145
A.2 AMPL Data for Double Feeder Assignment Problem .....	145
A.3 Triple Symmetric Similarity Matrix for Sample Data.....	146
A.4 AMPL Model for Triple Feeder Assignment Problem.....	147
A.5 AMPL Data for Triple Feeder Assignment Problem .....	147
A.6 Pseudo-codes for Min-Max heuristic .....	148
A.7 Pseudo-codes for SWAP heuristic.....	149
A.8 Hungarian Algorithm Example .....	150
A.9 Computation results for Data set 2 –R1 .....	152
A.10 Computation results for Data set 2 –R2 .....	153
A.11 Computation results for Data set 2 –R3 .....	154
A.12 Computation results for Data set 2 –R4 .....	155
A.13 Computation results for Data set 2 –R5 .....	156
A.14 Computation Results for Data set 3.....	157
B.1 Component Grouping Procedures from [72].....	159
B.2 PCB and component incidence matrix from [72].....	161
B.3 Component-to-component similarity matrix from Appendix B.2.....	162
B.4 Optimal Double feeder assignment for Appendix B.3 .....	162
B.5 Component and PCB grouping.....	163
B.6 Single feeder setup plan - IIGS .....	164
B.7 KTNS pseudo-code flowchart .....	165
B.8 Modified KTNS pseudo-code flowchart for double-feeder .....	167



## LIST OF APPENDICES (Continued)

<u>Appendix</u>	<u>Page</u>
B.9 Single feeder setup plan – IIGS/KTNS .....	169
B.10 Double feeder setup plan - IIGS/Modified KTNS and optimal assignment .....	170
B.11 Evaluation results by different grouping threshold values and methods on example data.....	171
B.12 Summary results and effects evaluation of industry data sets .....	172
C.1 2 <sup>4</sup> Factorial Experiment Design with one replicate and Results .....	174
C.2 Analysis of Variance for log(Feeder setup), Full model .....	175
C.3 Backward selection stepwise regression for log(Feeder setup).....	175
C.4 Analysis of Variance for CPU times, Full model.....	176
C.5 Analysis of Variance for CPU times, Reduced model .....	176
C.6 Backward selection stepwise regression for CPU times. ....	177

To my wife, Hui-I (鍾惠怡), and Forrest (李霖)

# **COMPONENT TO MULTI-TRACK FEEDER ASSIGNMENT AND BOARD SEQUENCING FOR PRINTED CIRCUIT BOARD ASSEMBLY**

## **CHAPTER 1 INTRODUCTION**

Life cycle for electronic products is shorter than ever as introduction of new technologies has become the norm. Competition in the electronic industry include both product design and manufacture, and the manufacturing of printed circuit boards (PCBs) is a major activity in virtually every segment of electronics manufacturing. Printed circuit boards are used in all kinds of electronic circuits, from simple one-transistor amplifiers to the biggest supercomputers.

### **1.1 Overview of PCB Assembly Process and Surface Mount Technology**

The technology for making PCB was invented in the 1930s and came into use during World War II. Before that time, circuits were constructed with point-to-point soldering; components mounted on an insulating board and interconnecting wires were hand soldered to the component leads. In the 1970s and 80s, pin-through-hole (PTH) components like DIP (dual-in-line package), integrated circuits (ICs) and leaded resistors were popular. Many designs are now switching to new, smaller surface-mount components and complicated, fine pitch leaded and high density packages such as ball grid array (BGA). Generally surface-mount packages are much smaller than equivalent through-hole packages, so they require smaller, more closely spaced pads, thinner traces, and more precise placement and soldering. Surface Mount Technology was developed to meet these requirements.

Surface Mount Technology (SMT) is increasingly being used in the design and manufacture of PCBs. The placement of surface mount components is a

critical task in the assembly of PCBs. The surface mount assembly process is a high-speed operation involving a series of steps performed by a pick-and-place machine. The primary inputs to the process are printed circuit boards, electronic components, and assembly control programs. The outputs from the machine are PC boards with components mounted on to locations specified through a control program. The components are supplied to the machine via feeder carriages in different package forms. Usually, PCBs are assembled by several assembly machines of different types and limited by the number of components and feeders the machine can hold.

At the start of a new production run, the appropriate assembly programs are loaded on to a surface mount machine's controller. The PCB is transferred by conveyor system to placement location. Location pins are usually used to register and secure boards. The positioning system adjusts the pattern coordinates of the PCB. The placement head is then moved to a designated position to pick a component. With the aid of a vision system, the component is moved to the board site and the placement position is adjusted. The component is then placed onto the board. The pick-and-place process is repeated until all required components are placed. The PCB is released and transferred to the another head or next station.

A typical pick-and-place machine has three control components: process control, machine control, and supporting databases. The process control component retrieves the data needed for the placement process from the database in real time. It also checks if pre-placement activities are required, as for example, preparation of new components. The machine control component is responsible for the control and monitoring of all activities associated with machine operation. This includes positioning of the carriage and the operation of tape feed and tape cutters on the feeders. Finally, the databases store the information required during a placement cycle. This includes information about the machine feeders, components, and placement programs.

## 1.2 PCB Assembly Systems

The SMT assembly process is commonly classified into three major types [62] according to the types of components used by PCBs. The first type contains only surface mount components. The second type contains both surface mount and PTH components but only surface are mount components mounted on the bottom side. Finally, the third type contains only discrete surface mount components glued to the bottom side. More than 90 percent of PCBs use combination of PTH and surface mount components and are often referred to as mixed assemblies. The process sequences and equipment are different for each assembly type. Figure 1.1 shows the typical process sequence involved in a single-sided PCB mixed assembly.

For different production volumes and requirements, there are different SMT equipment alternatives. Generally, these can be classified into four major categories according to the requirements for throughput and flexibility [62]:

- 1) High throughput
- 2) High flexibility
- 3) High throughput and flexibility
- 4) Low cost and throughput but with high flexibility

High throughput type SMT machines are designed to pick and place components using a servo-driven, cam-actuated, rotary turret placement head (Figure 1.2a). These machines are often referred to as high-speed placement machines or chip shooters, but only support the placement of components packaged in tape or in special magazines. The maximum throughput can vary from 10,000 to 60,000 parts per hour (pph). However, the actual throughput is dependent on component size, feeder type, feeder carriage movement, and the X-Y table movement. Some additional features can increase machine uptime, such as feeder bank that allows feeders to be loaded swiftly while the equipment is still in operation.

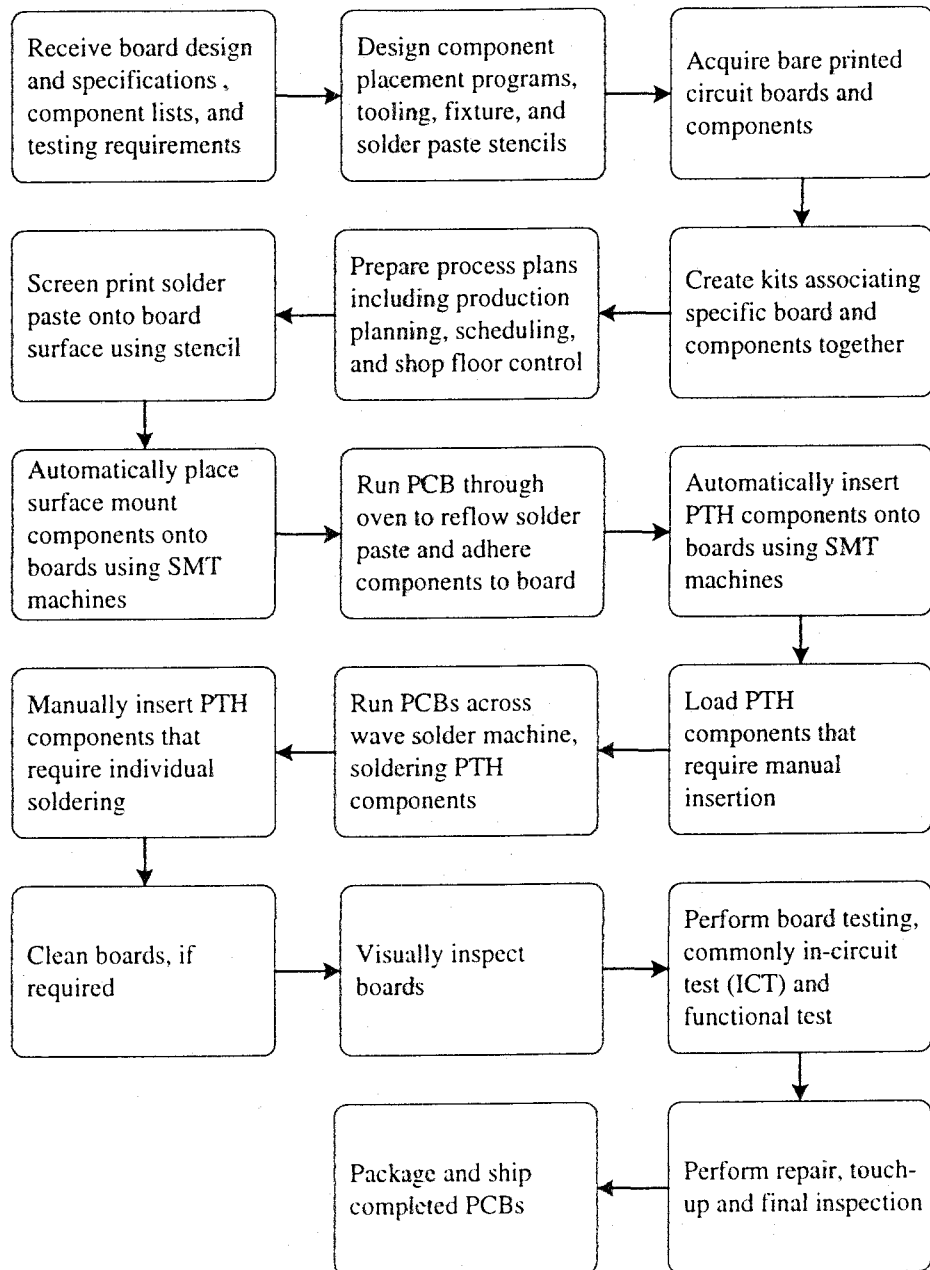


Figure 1.1 PCB assembly procedures

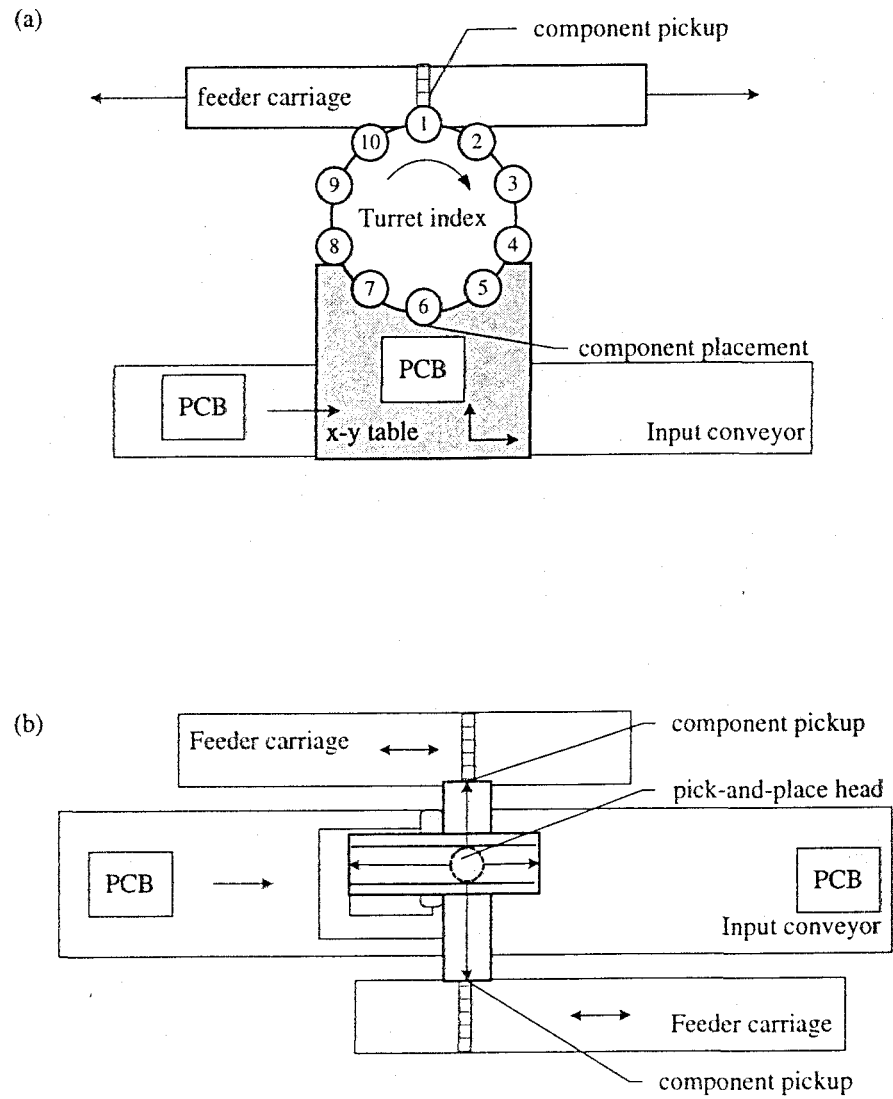


Figure 1.2 Pick-and-place machine with (a) rotary turret design (b) gantry design

High flexibility machines are also called multifunction placement machines and have lower throughput. The pick-and-place mechanism is based on an overhead gantry design (Figure 1.2b). The head will move to fixed feeder carriage position to pick a component, carry it to vision system for inspection, and then move to board site for placement. The placement speed is dependent on the component type, size, number of heads, feeder type and arrangement, carriage movement, and the distance from the feeder to the placement position.

In a production environment that needs high throughput and flexibility, a more common approach is to combine two types of machines in a line with complementary functions. The total throughput is dependent on the capabilities of individual machines and the assignment of components to machines.

A low cost and throughput but high flexibility machine is generally needed for prototype or laboratory work. Table 1.1 lists the comparisons between general high-throughput and high-flexibility machines [77].

Table 1.1 Comparison of general high-throughput and high-flexibility machines

	High throughput machine	High flexibility machine
Throughput (pph)	10,000 – 60,000	6,000 – 12,000
Component size (mm)	1.0×0.5 – 32×32	1.0×0.5 – 50.8×50.8
Placement accuracy (+/- mm)	0.1	0.02
Component lead pitch (mm)	0.65mm	0.4mm
Placement operation	Fixed pick and placement location, rotary turret head and moving X-Y table	Head moves between pick and placement locations
Carriage/Feeders operation	Carriage moves feeder to pick position	Carriage moves feeder to pick position
Feeder type	Tape-and-reel, bulk	Tape-and-reel, tube, matrix tray
8mm feeder slots	150/300 (w/ double feeder)	64



Surface-mount components are supplied to the end user in one of three configurations: bulk, tube magazine, and tape-and-reel. Tape-and-reel packaging is generally preferable for automation. For low-volume or prototype production, the tube magazine might be acceptable. Both tape-and-reel and magazine containers are clearly marked for efficient material control and easily adapt to the automated SMT placement equipment. Loose packaging of chip type components is less desirable because the part must be handled with special feeders for the assembly equipment.

Another consideration for selecting SMT machine is the number of feeders or the slot capacity. This number is a measure of the machine capacity to process an assembly with different part types. The more slots a machine has, the higher its capacity. The actual number of part types that can be accommodated, however, depends on the parts size and feeder types.

### **1.3 PCB Assembly Process Planning**

In PCB manufacturing, process planning plays a key role in determining productivity. As the degree of automation increases and more components are involved, the system becomes more complex. It is difficult, or even impossible, to manually arrange such a system. The relationships between the process planning, production planning, scheduling, and shop floor control, shown in Figure 1.3, are not purely serial or hierarchical. The decisions resulting from the interaction between them can be categorized into three levels [52]:

Level 1: (Grouping) Selection of machine groups and part families and assignment of families to groups;

Level 2: (Allocation) Allocation of components to machines when a group has more than one machine; and

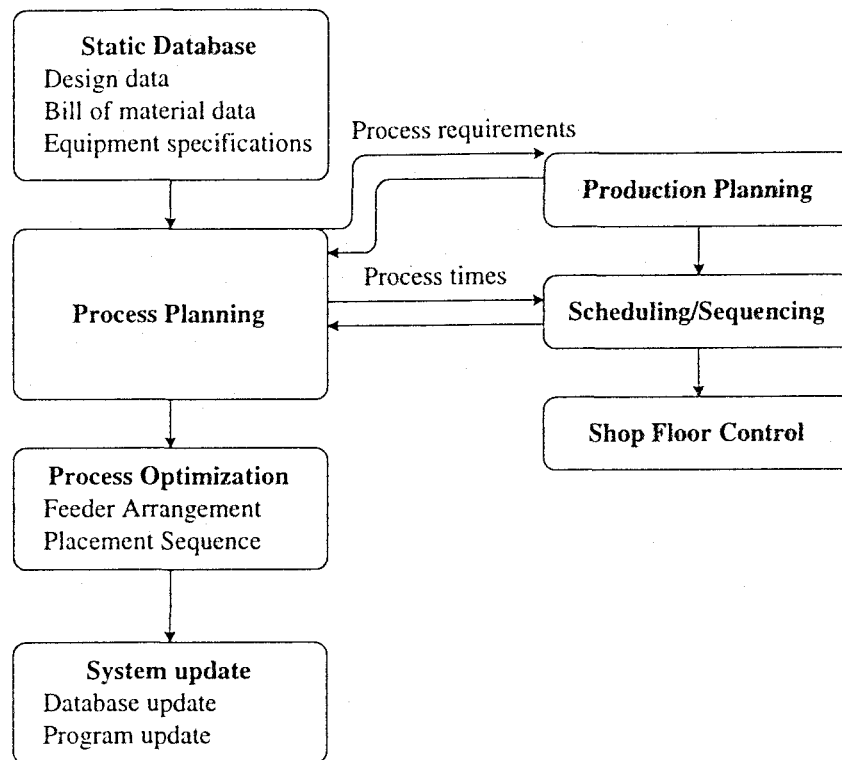


Figure 1.3 Relationships of process information and planning

Level 3: (Arrangement and Sequencing) Arrangement of component feeders and sequencing of placement operations for each machine and PCB.

With these decisions, the process optimization objectives can also be separated into three major strategies:

- 1) Minimization of process time: Arranging feeder locations on the carriage and searching for the shortest path of the placement sequence are two major tasks to satisfy this goal.
- 2) Minimization of setup time: Grouping component and board families and planning the board sequence can reduce the feeder changeovers.

- 3) Minimization of process and setup times: Simply combining strategies (1) and (2) may result in a conflict of objectives. To reduce setup time between two consecutive board assemblies by keeping the feeders in the same slots is contradictory to minimize the process time by rearranging feeder positions for fast pick and place movement. Thus, trade off between process time and setup time has to be made based on the production volumes and product variety.

Further review and discussion on optimization and setup strategy selection issues will be presented in Chapter 3.

#### **1.4 Research Objectives**

The motivation of this research originated from discussions with electronic companies which can be categorized as low volume, high mix operations. The research focus is on the analysis of setup management and the primary goal is to develop an optimization model for PCB assembly process with multi-track feeder configuration.

New technologies introduced in industry generate problems and provide new challenges. Before the introduction of multi-track feeders, reducing the setups of PCB assembly was based on single-feeder configuration in which only the allocation of the feeder was considered. However, if the machine can not hold all the parts required by the PCB board, either the board needs to be transferred to another machine which inserts the remaining parts, or the parts on the current machine need to be reconfigured. The multi-track feeder can increase the capacity of the SMT machine so that the machine can assemble more parts without re-configuration. However, the PCB assembly line will not benefit from the multi-track feeder by a poor setup plan. In fact, performance may deteriorate due to more complicated material management, allocation issues, and even human factors considerations.

An assignment plan for components to multi-tack feeders, along with board grouping and sequencing, would be expected to reduce the feeder changeovers.

The research methodology starts with the analysis of component families and board grouping. A board group is assumed to be assembled on the same machine with most common components. The multi-track feeder would accommodate two or more components. A model for the double-feeder, component assignment problem was first developed and then extended to the multi-track feeder, component assignment problem. The problem size is a major concern in the development of solution algorithms; the CPU time and the quality of results are used as the criteria in evaluating the efficiency and effectiveness of the algorithms.

The sequencing problem is next investigated in two ways: intra-group and inter-group. The goal is to divide the PCBs into groups with similar components so that a number of components can be handled on one machine. Finally, the multi-track feeder, component assignment is addressed to determine the board sequence to minimize the total number of feeder changeovers. The development of solution methodology involves application and integration of Group Technology, optimization, and heuristic algorithms. The algorithms are applied to data sets from industry. A random number generator is also developed to generate simulated data sets which are used for experimentation and verification.

## **1.5 Research Contributions**

This research develops a new methodology for addressing the component, feeder assignment and board sequencing problems. A generalized model for high-dimension symmetric assignment problem is developed. Heuristics incorporated with commercial MIP solver are implemented to approach efficient and near optimal solutions for large-size, low-dimension problems. Grouping technology is utilized to facilitate the solution of board sequencing problem. Previous compo-

nent/board grouping and sequencing methodologies are adopted and modified for improved solutions. Finally, a research approach is developed for combining component/multi-feeder assignment model and board sequencing problem.

The application is implemented on data sets from industry company. Experimentation on simulated data sets explores the characteristics of the application. Suggestions for applying the model developed in this research to different application areas are presented.

## **1.6 Organization of Dissertation**

Chapter 2 describes the problem background and context. An overview of the methodology and assumptions made in this research are also presented in this chapter. A background literature review is provided in Chapter 3. The mathematical modeling and heuristic development for component/multi-track feeder assignment problem are discussed in Chapter 4 followed by evaluation and experimentation. Chapter 5 presents the methodology for determining the board sequence and an approach to solve the overall problem. Application of the methodology to several industrial data sets and experimentation on simulated data sets are presented in Chapter 6. Finally, Chapter 7 concludes this dissertation with summary of results and a discussion of the limitations, tradeoffs and future extensions.

## CHAPTER 2 PROBLEM STATEMENT AND ASSUMPTIONS

This chapter first describes the application of multi-track feeders on SMT machines. This is followed by an example illustrating the problems to be addressed including the component-to-feeder assignment and the interaction of assignment with board sequencing. The two sub-problems and the overall problem definitions are presented along with the assumptions made in developing the solution approach. Finally, some comments are made about the current industrial practice.

### 2.1 Multi-Track Feeder Configuration

In the pick-and-place process, the surface mount components are installed on the feeders and then put on the carriage slots. Currently, multi-track feeder design is available to PCB assembly, which can accommodate more types of components and occupy less slot space. Feeders that can accommodate one type of component are called single feeders, and those can hold two types of component are double feeders (Figure 2.1). For example, a double feeder can hold two 8 mm tape and reel components.

The advantages of using multi-track feeders in SMT assembly machines include:

1. Increased machine input capacity so that large PCBs can be processed.
2. Decreased setup downtime because of insufficient parts on machine.
3. Reduced time to load components on to SMT machines, if planned appropriately.

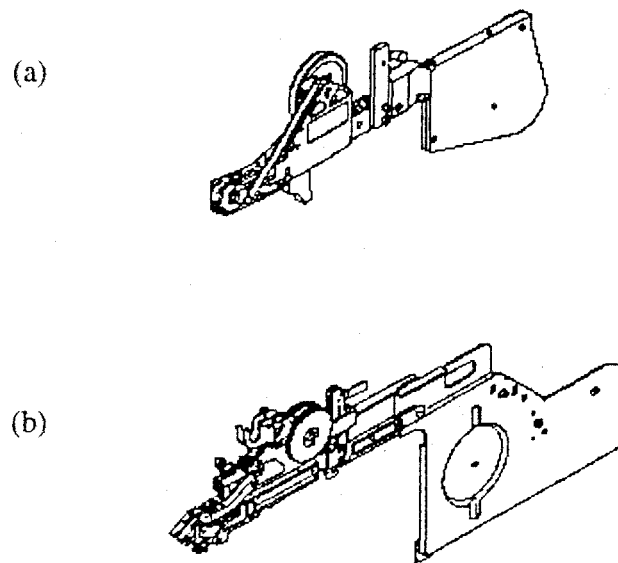


Figure 2.1 (a) A single feeder, and (b) A double feeder [13]

However, there are some drawbacks that discourage the use of multi-track feeders:

1. Increased time to install and uninstall components. This problem becomes more significant if configurations are frequently required in a low batch size and high product variety process.
2. More expensive than single feeders.
3. Increased effort in process planning, inventory management, and shop floor control.

## 2.2 Component-Multi-track Feeder Assignment Problem

Consider the following illustrative example. Table 2.1 shows a 0-1 incidence matrix of data for four PCBs that are to be produced using eight components. Cells

with “1” represent the use of a component in the production of a PCB and a “0” represents that a component is not used. For example, PCB 2 requires component 1, 4, and 8, and component 3 is only used by PCB 4.

Table 2.1 Example of a PCB incidence matrix

PCB	Components							
	1	2	3	4	5	6	7	8
1	0	1	0	0	1	0	1	0
2	1	0	0	1	0	0	0	1
3	0	1	0	1	1	0	1	0
4	1	0	1	0	0	1	0	1

In this research, a feeder setup is defined as one feeder being unloaded from carriage and replaced by another feeder that is ready for loading. Thus, a feeder setup will be required if a component is not available between two consecutive board assemblies.

### 2.2.1 Single Feeder and Setup Reduction

Notice that each PCB in Table 2.1 requires no more than four components and no less than three. Assume that an SMT machine has four slots so that all PCBs can be processed in one pass, and that only single feeders are to be used. Table 2.2 shows the single feeder setup with PCB sequence 1–2–3–4. Bold numeric text represents the component being used by the PCB currently being assembled; plain numeric text represents component that is not used but is installed on the carriage.



Table 2.2 Example of a single feeder setup

(A)					
PCB	Component				setups
1	2	5	7	6	0
2	1	8	4	6	3
3	2	5	4	7	3
4	1	3	6	8	4
total					10

(B)					
PCB	Component				setups
1	2	5	7	4	0
2	2	1	8	4	2
3	2	5	7	4	2
4	1	3	6	8	4
total					8

From the above example, it is clear that the feeder chosen to be replaced for the following assembly affects the total number of feeder setups. Table 2.2 shows that replacing component 2 with 1 for PCB 2 will result in an additional setup to put component 2 back on line when processing PCB 3. Table 2.2(B) also shows that including component 4 in the initial setup can save an additional feeder changeover.

### 2.2.2 Double Feeder and Setup Reduction

Now consider the double feeder case. If all the components can be used on double feeders, then, no matter how components are arranged and PCBs are sequenced, the system requires no feeder setup since the eight components are all loaded on the SMT machine. But what if there are only three feeder slots instead of four? The double feeders and components will require an arrangement plan because only six components can be loaded at any time. Assume that there are enough double feeders to hold all components and each feeder will occupy one slot. With the same PCB sequence, the double feeder setup is shown in Table 2.3.

Table 2.3 Example of double feeder setup with different assignments

assignment (A)				
(1,8) (2,7) (3,6) (4,5)				
PCB	Component			setups
1	(2,7)	(4,5)	(1,8)	0
2	(2,7)	(4,5)	(1,8)	0
3	(2,7)	(4,5)	(1,8)	0
4	(2,7)	(3,6)	(1,8)	1
total				1

assignment (B)				
(1,7) (2,4) (3,5) (6,8)				
PCB	Component			setups
1	(1,7)	(2,4)	(3,5)	0
2	(1,7)	(2,4)	(6,8)	1
3	(1,7)	(2,4)	(3,5)	1
4	(1,7)	(6,8)	(3,5)	1
total				3

assignment (C)				
(1,2) (3,4) (5,6) (7,8)				
PCB	Component			setups
1	(1,2)	(5,6)	(7,8)	0
2	(1,2)	(3,4)	(7,8)	1
3	(1,2)	(4,5)	(7,8)	1*
4	(1,2)	(3,6)	(7,8)	1
total				3*

\* Component 3,4,5, and 6 are reassigned to double feeder as (4,5) and (3,6)

In Table 2.3, eight components are grouped into four pairs and installed onto four double feeders. However, only three feeders are on the machine because the machine has three slots. Assignment (A) gives the best result since it has the lowest number of setups. Although both assignment (B) and (C) require the same number of setups, (C) needs a feeder reconfiguration between assembly PCB 2 and 3 because the four required components are installed on four different feeders and there is not enough spaces for all of them since there are only three slots. Reconfiguring a double feeder takes more time than just loading a feeder to the carriage. Besides, changing a component assignment may also require the system database to

be updated so that the SMT machine can locate the correct position of feeder and components for pick and placement.

The advantages and disadvantages shown in this double feeder example are summarized as follows:

1. Utilizing double feeders can increase the capacity of SMT machines so that larger PCBs can be assembled with one pass.
2. Assignment of components to double feeders significantly affects the advantages of using them. Poor assignment will not promote the benefits and possibly, hold off production.
3. Objective of minimization of process time by arranging feeder locations on carriages becomes more complicated due to the potential conflicts with the component/feeder assignment problem.

### **2.2.3 Problem Assumptions**

In the component-feeder assignment problem, the objective is to develop a mathematical model that can maximize the benefit of the multi-track feeder function. Since the issues involved in PCB assembly are interrelated, it is not realistic to solve all problems at the same time, not to mention that some problems have conflicting objectives. The following set of assumptions are developed based on visitations to PCB assembly lines, discussions with engineers, and experiences from working in the SMT laboratory in the Department of Industrial and Manufacturing Engineering at Oregon State University:

1. **Feeder can be mounted on any slot:** All feeder slots in a machine are of the same size, and any component feeder can be assigned to any feeder slot.

2. **Constant feeder loading and unloading time:** Feeders are loaded or unloaded one at a time, and the loading/unloading time is constant for all types of feeder. Preparation of feeders and installation of components are independent of loading and unloading.
3. **PCB design data are available:** The board-component incidence matrix contains the information of board requirements and component usage. This information can be obtained from PCB design data.
4. **Feeders and components are always available:** Feeders and components are assumed to be available in sufficient quantity through a production period. Inventory management and shop floor control are excluded from problem consideration.
5. **Total number of component types required by PCBs is greater than the SMT machine capacity:** This implicit precondition of the assignment problem, as shown in the double feeder setup example discussed above, states that if all types of components used by a group of different PCBs can be loaded on SMT machine at the same time, then there is no reason for solving the assignment problem since no feeder setup is required. However, it is not unusual that at least some PCBs require more component types than the machine can accommodate.
6. **PCB and component data are preprocessed:** Since component types that can be installed on multi-track feeder are limited (this also depends on the package format and machine's capability), the PCB design data are preprocessed and filtered so that only those PCBs with multi-track feeder-installable component are selected.

### 2.3 Multi-Track Feeder and Board Sequencing Problems

Reconsider the example in Table 2.3 with different PCB sequences and the feeder setup plans illustrated in Table 2.4. The table shows that the sequence of PCB assembly is also a factor to affect the feeder setup. Although assignment (A) performs better than assignment (B) in Table 2.3, their performance in terms of number of setups, is equivalent in Table 2.4 due to carefully planned sequence in (B) saving an additional setup.

Table 2.4 Example of double feeder setup with different PCB sequences

assignment (A)				
(1,8) (2,7) (3,6) (4,5)				
PCB	Component			setups
1	(2,7)	(4,5)	(1,8)	0
4	(2,7)	(3,6)	(1,8)	1
2	(2,7)	(4,5)	(1,8)	1
3	(2,7)	(4,5)	(1,8)	0
total				2

assignment (B)				
(1,7) (2,4) (3,5) (6,8)				
PCB	Component			setups
3	(1,7)	(2,4)	(3,5)	0
1	(1,7)	(2,4)	(3,5)	0
2	(1,7)	(2,4)	(6,8)	1
4	(1,7)	(3,5)	(6,8)	1
total				2

This example shows that the effort of searching for the “best” component-to-feeder assignment can be either augmented or diminished by the sequence of PCBs. A planned production sequence can compensate for the weakness of an inadequate assignment. Contrarily, a poor sequence can diminish the benefits of a good assignment. Another fact shown by this example is that there is no absolute opti-

mal assignment separate from an optimal sequence and vice versa. The interaction between the two affects the final results.

## 2.4 Overall Problem Statement and Assumptions

As the previous examples show, the objectives of this research are not only to search for an acceptable component-multi-track feeder assignment and a PCB sequence, separately, but also to develop an approach that looks at effective combination of both factors. The primary assumptions made in studying the overall problem are:

1. **High variety, low volume production:** The optimization objectives are different depending on the production environment, volume and product variety. For high volume, low mix production, minimizing process time will be more desirable than reducing setup time since a large proportion of total operation time is assembly time. On the contrary, a high mix, low volume production is usually dominated by setup time.
2. **Due dates are not included:** Production schedule will be based on a group of PCBs. Individual PCB scheduling within the group is not considered.
3. **PCB will be processed through the same SMT machine only once except double-sided design:** This precondition for sequencing problem implies that all PCBs are processed through a line of SMT machines. Components are distributed over machines according to their functions, capabilities, and workload to complete large PCBs. Double-sided design will be treated as two individual boards and given different identification numbers.

4. **Components can be loaded on the same machine multiple times but can not be duplicated on the line:** Although a SMT machine allows the duplication of the same component on a line due to high usage frequency, it is a waste of feeder space for a system attempting to increase the machine capacity in order to cover higher board variety.
5. **No interference between any two components:** The interference between any two components during assembly is caused by poor cooperation between design, production, and process planning. This issue is not related to the feeder setup, assignment, or the sequencing problem.
6. **PCB design factors are excluded:** Double-sided design will be considered as two different single-sided designs since the requirements of the process and the components may be different. However, from the point of view of assignment and sequencing problems, two PCBs using the same types of component are considered identical even though they are designed differently.
7. **Human factor considerations are excluded:** Human factor considerations are excluded from this research. For example, assigning components with similar part numbers together may be prone to operator mistakes. Time or costs caused by human error are not considered in the model.

## 2.5 Current Industry Application

The component-multi-track feeder assignment problem has been recognized to be significant in industry, and some planning applications have been developed to deal with it. Some focus on grouping of components according to their usage

frequency by PCBs, while others consider grouping by component similarity. For both methods, no estimate of effectiveness of the assignment is currently available. Moreover, there is a lack of application methodology and results that deal with assignment and sequencing problems together. One of the reasons is that the sequencing algorithms developed so far are based on single feeder configuration; most of these algorithms are not appropriate to be directly applied to the multi-track feeder case.



## CHAPTER 3 BACKGROUND AND LITERATURE REVIEW

Production in the electronics industry, especially the assembly of printed circuit board, has been thought to be a typical mass-production process for some time. Numerous research studies, focused on various aspect of PCB industry to improve productivity and quality, have been reported in literature. This chapter discusses the research background and reviews related work pertinent to the proposed research.

### 3.1 Trade-off Between Process Time and Setup Time

Component placement and machine setup are the two major activities in PCB assembly system. The relationship of process time and setup time to production determines the selection of the optimization strategy. McGinnis *et al.* [52] assessed different system characteristics and proposed setup strategies for the feeder arrangement problem to minimize the movement distance, and for placement sequencing problems to minimize the time spent on setups, as well as the component allocation problem to balance the machine workloads.

A typical assembly process involves a series of assembly machines. Two major categories of machines are distinguished by the mechanism used to perform operations. Sequential assembly machine performs sequential pick-and-place operations, and a cycle consists of a retrieval and placement of a single component. A concurrent assembly machine is equipped with specialized tools or multi-head so that more than two operations may be performed concurrently

If the machines are connected by transportation devices, the PCBs typically will be processed in an order following the conveyor system. McGinnis *et al.* [52] referred to the conveyor-linked machines as coupled, and a system with batch transfer as decoupled. Setup of a single machine in coupled system effectively

idles the entire system. In contrast, a decoupled system may allow other machines to continue production during the setup of a single machine.

Ji *et al.* [36] introduced linear assignment-based approach for planning component placement sequences with fixed boards and fixed feeders. The implementation is divided into two elements, namely *construction of cost matrix for the assignment algorithm* and a *merge operation for generating the final placement sequence*.

Askin *et al.* [3] proposed three heuristics for component assignment and group formation on a decoupled system: component-assignment/work-load balancing algorithm, work-load balancing algorithm with shortest total processing time (SPT), and natural board subfamily algorithm. The objective is to minimize the makespan for assembling a batch of boards with a secondary objective of reducing the mean flow time.

Peters and Subramanian [61] proposed four strategies placing different importance on the optimization of processing time versus setup time (Figure 3.1).

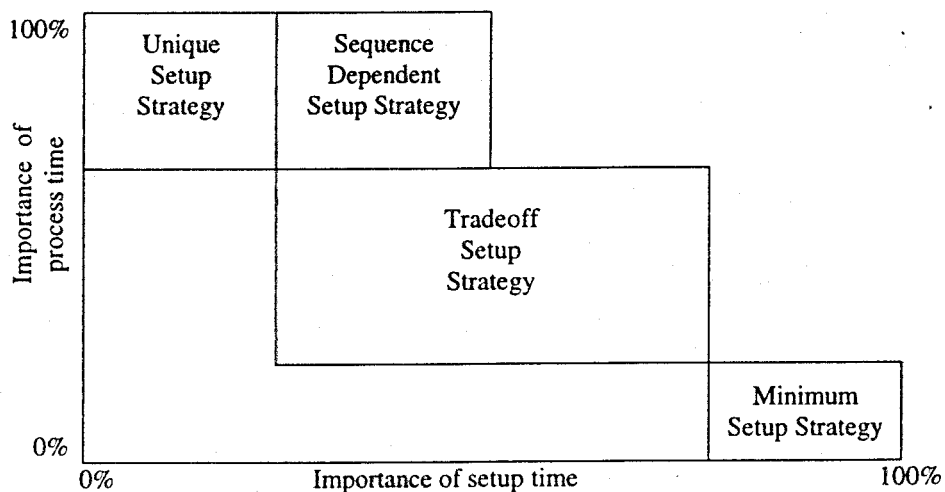


Figure 3.1 Optimization strategy and the importance of process versus setup times.

1. Unique setup strategy tends to be used in high product volume, low product variety applications since it emphasizes the processing time of each product. This strategy determines the machine state and placement sequence independently for each product.
2. Sequence dependent setup strategy uses a fixed machine state and placement sequence for each product and then determines the allocation of products on each machine to minimize the makespan. The process time is fixed and the product allocation and sequencing decisions only affect the setup time.
3. Minimum setup strategy chooses the machine state for processing a product to minimize the setup time from the current machine state. That is, the system only performs the required setup for a product on a machine. This strategy is typically used in situations with low product volume but high product variety.
4. Tradeoff setup strategy is intended to select specific optional setups that balance the tradeoff between processing time and setup time, hence it is applicable in medium volume, medium variety situations.

The process time for a PCB basically is a function of the total number of components and the time to place a particular component by the machine. The actual process time is dependent on the placement sequence and feeder locations of the components. Two types of latency [23,26], feeder latency and board latency, may increase the process time. During the assembly process, the feeder carriage needs to move to the appropriate location for component retrieval, and the speed of carriage is relatively slow compared to the pick-and-place device movement. The largest feeder distance between two placements that can be moved within the placement time is typically referred to as free feeder distance. If the machine has to wait for the carriage movement, feeder latency occurs. An equation developed by

DePuy [23] may be used to estimate the feeder latency time that is the total feeder latency distance divided by the feeder speed. The feeder latency distance is the total feeder movement distance adjusted by the total free feeder distance. Moyer and Gupta [55] developed a feeder slot allocation heuristic to minimize feeder travel distances by utilizing a distance matrix to identify the distance between components. However, the component placement sequence is fixed, and the previous or future PCB assemblies are not taken into consideration.

The board latency occurs when the machine is waiting for the X-Y table or pick-and-place head to move to the appropriate position for component placement. The objective of most research to date regarding this problem is to minimize the movement distance between the retrieval position and the placement position, i.e., to minimize the cycle time. The factors to be considered include the position of feeder and the retrieval and placement times. Component placement sequence is involved to reduce the board latency. The SMT systems are now equipped with software to estimate the process time of a particular board according to the machine setup. Ellis [26] developed an estimator incorporated with the machine software to estimate the process time.

The travelling-salesman problem (TSP) formulation is widely used in research to model the distance-minimization problem. Moyer and Gupta [57] developed a heuristic that determines the component placement path based on the component coordinates with a moving X-Y positioning table and a turret type placement mechanism.

### **3.2 Group Technology**

The Group Technology (GT) concept has been successfully applied in cellular manufacturing [6,7,12,19,31,41]. The PCBs and components may be considered analogous to parts and machines and grouped as in cellular manufacturing

system [11,22,32,48,71,72]. Many methods have been developed to apply the GT concept [20,21,33]. *Clustering analysis* is one of the most frequently applied mathematical tools in GT. King [41] explored the existing cluster analysis methods like the *single linkage cluster analysis* and the *bond energy method*, and developed a new approach called the *rank order clustering* method.

The single linkage cluster analysis was developed by Sneath [69] and was applied to group analysis in production by McAuley [50]. The method involves a hierarchical process of machine grouping in accordance with computed similarity coefficients [54,68]. The disadvantage is that once the machine groups have been formed it is necessary to assign the components to them.

McCormick *et al.* [51] developed the bond energy method and stated that the problem of clustering a machine-part matrix by permuting rows and permuting columns could be interpreted as a pair of independent quadratic assignment problems. For example, a bond is said to exist (see 3.2) between two adjoining column elements  $a_{14}$  and  $a_{15}$  in the incidence matrix (3.1). The goal is to rearrange the row and column position and maximize the total bond energy.

$$A_{4 \times 5} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} & 1 & & 1 & 1 \\ 1 & & 1 & & \\ & 1 & & 1 & \\ 1 & & 1 & & \end{bmatrix} \end{matrix} \quad (3.1)$$

$$A_{4 \times 5} = \begin{matrix} & \begin{matrix} \text{Column bond energy} \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 1 \\ 1 \\ 1 \\ 1 \end{matrix} & \begin{bmatrix} & 1 & & 1 & 1 \\ & & 1 & & \\ & 1 & & 1 & \\ 1 & & 1 & & \end{bmatrix} \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \end{matrix} \begin{matrix} \text{Row bond energy} \\ \\ \\ \end{matrix} \quad (3.2)$$

The *rank order clustering* method (ROC) was developed by King [41] to group machine and component families. The machine-component incident matrix is read by row and column 0-1 vectors as a binary word. For example, the first row vector in (3.1) can be represented by a binary number, 01011, and converted to the decimal number, 11. The matrix is rearranged by the rank order of column and the rank for the rows is recalculated. The process is repeated until the rank order does not change. Unlike single linkage cluster analysis, which after machine grouping require a secondary process of component allocation to the machine groups, the ROC performs both tasks together.

Kusiak [44] compared the matrix model, the *p-median* model and the classical group technology concept. There are two basic formulations of the clustering models: matrix and integer programming formulation. In the matrix formulation, judgement regarding the number of clusters and the numbers of elements in each cluster is performed by a human, while in the integer programming formulation, both of them are determined by the clustering algorithm. The clustering analysis starts from a machine-part incidence matrix and rearranges rows and columns. A new matrix is identified from matrix (3.1) with two visible families of machines and parts.

$$A_{4 \times 5} = \begin{matrix} & \begin{matrix} 1 & 3 & 2 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 4 \\ 1 \\ 3 \end{matrix} & \begin{bmatrix} 1 & 1 & & & \\ 1 & 1 & & & \\ & & 1 & 1 & 1 \\ & & 1 & 1 & \end{bmatrix} \end{matrix} \quad (3.3)$$

This matrix formulation is difficult to represent and to visualize clusters when the incidence matrix gets large, and in most cases it is hard to obtain diagonal or close to diagonal structure of the clustered matrix.

The  $p$ -median model transforms the incidence matrix  $A_{ij}$  into individual 0-1 vectors that represent the relationship for each part and machine. The similarity is thus defined as:

$$p_{ij} = \sum_{k=1}^m \delta(a_{ik}, a_{jk}), \text{ where } \delta(a_{ik}, a_{jk}) = \begin{cases} 1, & \text{if } a_{ik} = a_{jk} \\ 0, & \text{otherwise} \end{cases} \quad (3.4)$$

The similarity matrix  $P_{5 \times 5}$  for (3.1) is:

$$P_{5 \times 5} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 4 & 3 \\ 4 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 3 \\ 1 & 3 & 1 & 3 & 0 \end{bmatrix} \end{matrix} \quad (3.5)$$

The problem can then be formulated by an integer program (IP) to maximize the total sum of similarity:

$$\begin{aligned} &\text{Maximize} && \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_{ij} && (3.6) \\ &\text{Subject to} && \sum_{j=1}^n x_{ij} = 1, \text{ for all } i = 1, \dots, n \\ &&& \sum_{j=1}^n x_{jj} = p \\ &&& x_{ij} \leq x_{jj}, \text{ for all } i = 1, \dots, n \text{ and } j = 1, \dots, n \\ &&& x_{ij} \in \{0, 1\} \quad \text{for all } i, j \end{aligned}$$

Solving the above IP results in part families and the corresponding machine cells. This model can be used for general grouping problems. Chen *et al.* [18] developed a similar IP model for production planning of flexible manufacturing systems. Cheng *et al.* [20] formulated a 0-1 quadratic programming model solved by a truncated tree search algorithm to group machines.

### 3.3 Assignment Problem

Assignment problems are generally referred to as those problems that seek the best “match” or “group” configurations to approach some objective. The assignment problems belong to the combinatorial problem, i.e., a type of discrete optimization problem. They can be categorized into three groups:

#### 1. Unbalanced Asymmetric Assignment problem (UAA)

Consider a set of machine  $Q = \{Q_1, \dots, Q_m\}$  and a set of jobs  $R = \{R_1, \dots, R_n\}$ , where  $m$  and  $n$  are not equal. Define  $c_{ij}$  as the cost coefficient of assigning  $R_j$  to  $Q_i$ ,  $s_{ij}$  as the resource required by machine  $Q_i$  to perform job  $R_j$ , and  $b_i$  as the resource capacity of  $Q_i$ . The assignment problem is to assign each job to exactly one machine such that the total resource requirement of the jobs assigned to each machine is within the capacity of that machine and the total cost is minimized. This assignment problem can be formulated as a pure 0-1 integer linear program:

$$\begin{aligned}
 &\text{Minimize} && \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} && (3.7) \\
 &\text{Subject to} && \sum_{i=1}^m x_{ij} = 1, && \text{for } j = 1 \text{ to } n \\
 &&& \sum_{j=1}^n s_{ij} x_{ij} \leq b_i, && \text{for } i = 1 \text{ to } m \\
 &&& x_{ij} \in \{0, 1\}, && \text{for all } i, j
 \end{aligned}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}$$

#### 2. Balanced Asymmetric Assignment problem (BAA)

Consider the above problem except that  $m$  and  $n$  are equal in this case, and there is no resource limitation. The pure 0-1 integer linear programming formulation is:



$$\text{Minimize} \quad \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (3.8)$$

$$\text{Subject to} \quad \sum_{i=1}^n x_{ij} = 1, \quad \text{for } j = 1 \text{ to } n$$

$$\sum_{j=1}^n x_{ij} = 1, \quad \text{for } i = 1 \text{ to } n$$

$$x_{ij} \in \{0, 1\} \quad \text{for all } i, j$$

where

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}$$

This assignment problem is also known as the *bipartite matching* problem that finds a *complete*, or *perfect*, matching [28,46,58,59]. Each job and machine will be assigned only once. However,  $x_{ij}$  is not equivalent to  $x_{ji}$ . This integer-programming model can be solved directly using commercial computer optimization packages. However, with large problems, the computation run time is prohibitive, as for many practical production planning problems. A primal and dual algorithm called Hungarian method developed by Kuhn [42] solves BAA problem optimally and efficiently. The Hungarian method is further improved by Wright [79]. A primal method which is “dual to” the Hungarian method is developed by Balinski [5] for the assignment and transportation problems. Another matrix-based algorithm for the BAA problem is proposed by Ji *et al.* [35], which is similar to the dual simplex method.

### 3. Balanced Symmetric Assignment problem (BSA)

Consider  $n$  objects that must be grouped into  $n/2$  pairs. Every object has to be in exactly one pair and each pair must consist of two distinct objects. This grouping problem is known as *symmetric assignment* problem, which is similar to BAA problem except that an additional restriction is imposed:

$$x_{ij} = x_{ji} \quad \text{for all } i, j \quad (3.9)$$

This restriction forces the assignment to be symmetric, that is, each pair is distinct. Note that only square matrix (same dimension in row and column) can have symmetric assignment.

The symmetric assignment problem was originally formulated by Murty [58]. Devine [25] developed a relatively simple branch and bound algorithm for the symmetric assignment problem to find the pairing of oil wells in order to minimize the total drilling cost; the size of problems ranged from 20 to 40.

### *3.3.1 Combinatorial Optimization*

Another issue in the assignment problem is optimization. Optimization problems can be divided into two broad categories [58]: those with continuous variables, that is, they can assume all possible values within their range of variation; and those with discrete variables, in which some or all the variables are restricted to assume values within specified discrete sets. The general linear programming (LP) problems belong to the continuous type, and the combinatorial problems are the discrete class of optimization. Although not all combinatorial optimization problems can be easily described mathematically, in general, they can be formulated as Integer Programming (IP) problems which are more difficult to deal with than the LP problems. There are two classes of IP problems: Pure IP, in which all decision variables are restricted to only integer values; and Mixed IP, in which there are mixed continuous and integer decision variables. In each class, there are two subclasses: 0-1 integer decision variables and general non-negative integer decision variables.

The methods that can solve combinatorial optimization problems include exact algorithms and heuristic algorithms. Exact algorithms solve problems optimally though the solution time may be unacceptable. On the other hand, heuristic algorithms solve problems efficiently but do not promise optimal solution quality.

A problem's complexity is one factor that determines whether exact or heuristic algorithms should be applied.

### 3.3.2 *NP-Complete Class Problem*

Time complexity is the measurement for an algorithm's computational capability. An algorithm is called polynomial-time algorithm if the required number of operations grows as a polynomial pattern with the size of the input. On the other hand, an algorithm is called an exponential-time algorithm if it is not of polynomial order. It is obvious that an exponential function grows much faster than polynomial function as problem size increases.

If a problem can be solved by a known polynomial-time algorithm, then it is defined to be in the complexity class  $P$ . A problem is defined to be a  $NP$  (non-deterministic polynomial) class if a given solution  $x$  is a "yes-no" instance and can be verified by a polynomial-time algorithm. Furthermore, a class of  $NP$ -complete problem is the "most important" problems in  $NP$ . Solving a  $NP$ -complete class problem requires a computational time that grows (in the worst case) exponentially with problem size. Many combinatorial optimization problems such as the well-known Travel Salesman problem (TSP) and the general 0-1 integer programming problem belong to the  $NP$ -complete class [40].

Since the assignment problem is a general 0-1 integer-programming problem, it is also a  $NP$ -complete class problem, which means the difficulty of the problem will grow as the problem size increases. Thus, a proper strategy or methodology is required to solve  $NP$ -class problem.

### 3.3.3 *Exact and Heuristic Algorithm*

Although *NP*-complete class problems strongly suggest that there are no efficient algorithms for solving them optimally, it does not mean there is no *exact algorithm* that can be used. The factor is the problem size. Only small to medium sized problems can usually be solved efficiently. However, there are some exceptions; even though the problem is a *NP*-complete class, there may be special cases in *P* class which can be solved efficiently, such as the branch-and-bound algorithm, dynamic programming techniques, or methods based on the theory of linear programming. Two issues are generally encountered when applying these exact algorithms to *NP*-complete class problems:

1. Exact algorithms can not be expected to solve a *NP*-complete problem optimally in a reasonable amount of computation time.
2. The computer resources like memory requirements impose limits on the use of exact algorithms.

These two issues interact because some exact algorithms require an increasing amount of memory as the algorithms progress. As more computer memory is used, the computation speed slows down. In most cases, the solution progress is aborted because CPU time is too large or memory is exhausted. Therefore, a heuristic algorithm that solves for near-optimal solutions is commonly developed for better efficiency and less computer resource requirements. The disadvantage of using heuristic algorithm is that there is no easy way of telling how close the solution is to optimality. Table 3.1 lists some of the commonly used heuristics [67]. Heuristic algorithms may be designed to be flexible in dealing with complex real-world problems, easy to design and implement, and to be used in combination with exact algorithms.

Table 3.1 Comparison of commonly used heuristics

Heuristic	Efficiency	Solution Quality	Methodology
Greedy Heuristics	best	worst	Generates a solution by considering each solution component one at a time until a feasible solution is constructed.
Neighborhood search or Local search	fair	fair	Begins with an initial feasible solution and improves it by a sequence of exchanges in a neighborhood search. It can be combined with Greedy heuristics to obtain an initial solution.
Mathematical relaxation	worst	best	Relaxes some constraints of the original problem to yield a solvable relaxed problem. A solution to the relaxed problem is then used as initial solution to obtain a feasible solution for original problem.

### 3.4 PCB Sequencing Problem

The PCB assembly is a sequence-dependent production issue. The sequence of PCBs significantly affects the setup time. Due to limited machine capacity, the components are required to be loaded/unloaded between two PCB assemblies. Therefore, two similar PCBs are likely to be sequenced together to reduce the component changeovers. Sequencing problem also arises in different manufacturing areas that require different tools for multiple jobs.

Carmon *et al.* [14] divided PCB assembly sequencing into two large categories: sequence-dependent scheduling (SDS) and group setup (GSU). In the GSU method, PCBs are assembled in two stages, the common component stage and the residual component stage. In other words, each component can be loaded only once, but each PCB (job) can be loaded more than once. Theoretically, the GSU method always generates the least total setup time for production plans. However, in real practice, this method is difficult to implement because it requires complete control on the production schedules and when lot sizes are fairly large, the long production makespan generated by this method may be fatal. In SDS, setup is

based on the batch sequence. Before a new batch is assembled, all components must be in the feeders. That is, the PCB loading is restricted to once only, but each component can be loaded more than once. This is a more realistic production practice.

Hashiba and Chang [32] introduced a systematic and practical method to the setup reduction problem in the SDS category. Their goal is to reduce setups for PCB assembly machines by improving the assembly sequence. An integer programming formulation was developed to minimize the number of total setups; heuristics were then developed for solving the model. The problem is decomposed into three parts:

1. Grouping PCB types into some groups, which is based on the component commonality among different PCB types. No setup is necessary when changing from one PCB type to another in each PCB group. The effective number of PCB types can be reduced from the number of real PCB types to the number of PCB groups.
2. Ordering the PCB groups, which determines the assembly order of PCB groups. Since no setup is needed in each group, the assembly order in a group can be arbitrary. On the other hand, the assembly order among groups has significant effect on the number of setups.
3. Assigning setup component types for each job, i.e., the order in which the component types should be on the feeders to minimize the total setups. Considerations include the unused components left unchanged from previous group.

Maimon and Shtub [48] combined SDS and GSU methods and formulated a mixed-integer non-linear programming model to solve the set-up reduction problem. There are two interrelated problems to be solved simultaneously:

1. Each PCB should be loaded with at least one group of PCBs and components on to the assembly machine
2. Each component should be loaded with at least one group of PCBs and components on to the assembly machine.

The two problems interact since each PCB should be loaded on the machine with a subset of the groups such that each of the components required for that PCB is a member of at least one of the groups in the subset. The objective function is to minimize total set-up time of PCBs and components. In addition, in the PCB grouping problem the sets are subject to the capacity constraint of the assembly machine, i.e., the maximum number of components that can be loaded simultaneously on to the machine.

The sequencing problem is frequently viewed as a Travelling Salesman Problem (TSP) whose objective is to minimize the travel distance among cities. The traditional techniques for an exact solution are branch and bound, dynamic and integer programming. Randhawa *et al.* [63] utilized branch and bound technique to evaluate the mean setup time in sequencer scheduling combined with an integer programming model to minimize the total sequencing costs. Tang and Denardo [75,76] presented a non-LP-based branch-and-bound procedure to deal with a job scheduling problem for a flexible manufacturing machine. They also developed an optimal tool replacement policy called *Keep Tool Needed Soonest* (KTNS) to minimize the total number of tool switches for a specific job sequence.

Because of computational complexity, those exact algorithms have been found impractical for large problems. Therefore, the use of heuristics becomes necessary for industrial problems. Charles-Owaba [17] developed a setup time matrix representing mathematical relation between machine setup times and similarity of parts. The objective is to derive a sequence that minimizes the machine

setup time for a set of parts. Five heuristics for solving this problem were compared.

Genetic algorithm has also been used to solve the sequencing problem. Starting with a population of random sequences encoded and decoded as bit streams, the algorithm performs crossover, mutation and regeneration processes to evolve a new and better sequence from the parent population. The new offspring rejoins the population and replaces the inferior member in order to repeat next evolving session. Maimon and Braha [49] presented a genetic algorithm approach to the component switching problem. They use KTNS policy to evaluate the quality of offsprings.

Grouping Technology is frequently used to group the jobs or PCBs before the sequencing process. Sule [71] developed a tabular method for machine grouping and job allocation in cellular manufacturing. This method is then used in [72] to group components and PCBs. The PCBs are then sequenced according to the then-current machine setup within and between groups.

### 3.5 Interrelated Problems

The complexity of the PCB assembly production arises from the interrelated decisions of the individual problems. Interactions and conflict of objectives make the problems more difficult to solve. Generally, grouping the components and PCBs can reduce the problem complexity. Balancing the tradeoffs is required to resolve the interactions and conflicts. Research studies focused on developing an integrated approach are summarized below.

Ammons *et al.* [1] developed a mixed integer programming model of the component allocation problem for the unique setup strategy, family setup strategy, and a variation of the decompose and sequence strategy. This model is extended by



DePuy [23] to include feeder assignment issues. Branch and bound algorithm is used to solve the model for both the unique and family setup strategies.

Moyer and Gupta [56] proposed a heuristic algorithm, referred to as the acyclic assembly time algorithm, to minimize the system time required to place components on a circuit board. The algorithm determines an initial process plan for component placement sequence and feeder allocation from the given input, and then uses a recursive approach to improve the initial solution.

Sanders [65] investigated operational planning for PCB assembly based on a medium-demand, medium-variety production on a mixed-model assembly line with two identical machines. Given weekly PCB requirements, the objective is to maximize the number of PCB types produced daily. The operational plan determines the subset of PCBs to be produced daily (part selection problem), sequences the PCBs (sequencing problem), and assigns each PCB type's components to the machines (workload balancing problem). A mixed integer linear programming model is developed. Each sub-problem is solved by heuristic procedures.

Lach [45] focused on minimizing the cycle time for multiple PCB types without set-ups by determining the ordering of components in the SMT machine feeder carriage, which leads to the lowest production time after the subsequent component placement sequence optimization problem has been solved.

Takvorian [74] developed a dynamic programming formulation and an approximation algorithm for solving the PCB sequencing problem. Jiang [37] dealt with PCB assembly in a multi-stage, multi-line, and multi-product system. A shop floor scheduling methodology called Repetitive Flexible Flow Lines (RFFL) is proposed to solve five decomposed sub-problems: (1) Daily product mix and lot sizing, (2) Optimal transfer batch determination, (3) Global sequencing with sequence dependency of setups, (4) Local stage dispatching, and (5) Dynamic events response.

Hillier [33] developed an optimal solution technique to assign the PCBs and components to the machines and a manual process so as to minimize cost for the single machine case and for the multiple machine case where boards are not allowed to be set up on more than one process. For multi-process cases, the tasks of minimizing and balancing the machine workloads are performed simultaneously.

## CHAPTER 4 COMPONENT TO MULTI-TRACK FEEDER ASSIGNMENT PROBLEM

In this chapter, the double feeder setup example introduced in chapter 2 will be discussed and used to illustrate the development of a methodology for the assignment problem. Integer programming (IP) models are formulated for the problem. Limitations of IP models and application are discussed. Several heuristics are developed to compensate for these limitations; these are then evaluated and compared.

### 4.1 Methodology

As discussed in chapter 2, the component to multi-track feeder assignment problem interacts with the board sequencing problem. A planned PCB sequence can reduce feeder setups and avoid reconfigurations as in the assignment shown in Table 2.3(C) (and reproduced in Table 4.1C). Therefore, the component-feeder assignment problem should be dealt with before performing PCB sequencing.

Since optimal assignment does not exist in isolation, it is appropriate to define an objective to measure the quality of assignment. Re-examine the PCB incidence matrix in Table 4.1. Notice that PCB 3 and 4 both require four components. Those components are better not to be assigned to four separate double feeders, otherwise reconfiguration is unavoidable. The assignment (C) shows that all pairs of components assigned to the same double feeder are not used by the same PCB. Comparing this to the assignment (A) and (B), it can be observed that if the common components are put together, the assignment will be more effective and avoid feeder reconfiguration.

Table 4.1 Example of PCB incidence matrix and double feeder assignments reproduced from Table 2.1 and 2.3

PCB	Components							
	1	2	3	4	5	6	7	8
1	0	1	0	0	1	0	1	0
2	1	0	0	1	0	0	0	1
3	0	1	0	1	1	0	1	0
4	1	0	1	0	0	1	0	1

Double feeder assignment (A) (1,8) (2,7) (3,6) (4,5)				
PCB	Component			setups
1	(2,7)	(4,5)	(1,8)	0
2	(2,7)	(4,5)	(1,8)	0
3	(2,7)	(4,5)	(1,8)	0
4	(2,7)	(3,6)	(1,8)	1
total				1

Double feeder assignment (B) (1,7) (2,4) (3,5) (6,8)				
PCB	Component			setups
1	(1,7)	(2,4)	(3,5)	0
2	(1,7)	(2,4)	(6,8)	1
3	(1,7)	(2,4)	(3,5)	1
4	(1,7)	(6,8)	(3,5)	1
total				3

Double feeder assignment (C) (1,2) (3,4) (5,6) (7,8)				
PCB	Component			setups
1	(1,2)	(5,6)	(7,8)	0
2	(1,2)	(3,4)	(7,8)	1
3	(1,2)	(4,5)	(7,8)	1*
4	(1,2)	(3,6)	(7,8)	1
total				3*

\* Component 3,4,5, and 6 are reassigned to double feeder as (4,5) and (3,6)

#### 4.1.1 Example Illustration

To assign common components together, the symmetric matrix in Table 4.2 is constructed from the incidence matrix in Table 4.1 to measure the degree of similarity among parts. The numbers in this matrix represent the number of PCBs using both the corresponding components. For example, the similarity value for component 2 and 5 is two because they are used by PCBs 1 and 3.

Table 4.2 Similarity matrix for double-feeder example

Component	1	2	3	4	5	6	7	8
1	-	0	1	1	0	1	0	2
2	0	-	0	1	②	0	2	0
3	1	0	-	0	0	1	0	1
4	1	1	0	-	1	0	1	1
5	0	②	0	1	-	0	2	0
6	1	0	1	0	0	-	0	1
7	0	2	0	1	2	0	-	0
8	2	0	1	1	0	1	0	-

For ease and consistency of representation, the following notation is used in this chapter:

- $m$  total number of PCB types;
- $n$  total number of components;
- $r$  number of feeder tracks;
- $A$   $(a_{ij})_{m \times n}$ , the PCB incidence matrix;
- $a_{ij}$   $ij$ th element of matrix  $A$ ;  $a_{ij} = 0$  or  $1$ ;
- $P$   $(p_{ij})_{n \times n}$ , the component-component similarity matrix;
- $p_{ij}$   $ij$ th element of matrix  $P$ , similarity value between component  $i$  and  $j$ ;
- $p_{i\bullet}$  all similarity values associated with component  $i$ ;
- $x_{ij}$  decision variable, 1 if component  $i$  and  $j$  are assigned together, 0 otherwise;
- $x_{\bullet j}$  all decision variables associated with component  $j$ .

According to the similarity matrix in Table 4.2, the similarity values for the three assignments in Table 4.1 are displayed in the following table:

Table 4.3 Similarity measurements for double-feeder example

	Assignment	calculation	Similarity value
(A)	(1,8) (2,7) (3,6) (4,5)	$p_{18} + p_{27} + p_{36} + p_{45}$	$2 + 2 + 1 + 1 = 6$
(B)	(1,7) (2,4) (3,5) (6,8)	$p_{17} + p_{24} + p_{35} + p_{68}$	$0 + 1 + 0 + 1 = 2$
(C)	(1,2) (3,4) (5,6) (7,8)	$p_{12} + p_{34} + p_{56} + p_{78}$	$0 + 0 + 0 + 0 = 0$

This shows that assignment (A) has the largest similarity value whereas assignment (C) has none. The outcome is consistent with the results in Table 4.1 in terms of number of feeder setups.

There are two findings from this illustration. First, it shows that, considering only the assignment problem, the higher the similarity value, the less the number of feeder setups. The similarity value is thus an effective and logical measurement for assignment with the objective of maximizing the sum of similarity values. Second, components with high similarity values are usually associated with PCBs using more common parts. If similar boards can be grouped and sequenced, then it is possible to further reduce feeder setups. This is valuable information that provides a connection between the assignment and sequencing problems.

To extend the discussion to the multi-track feeder case, some basic properties of the similarity matrix are summarized:

1. Symmetry property, the relationship of  $p_{ij}$  is the same as  $p_{ji}$ ;
2. Diagonal matrix is non-number (i.e., does not exist);
3. Matrix dimension is  $r$ , the number of feeder tracks; and
4. Matrix size is  $n^r$ .

### 4.1.2 Methodology Description

Many applications have used the assignment problem as a mechanism for representing and solving them. Kusiak [44] used a similar method to deal with the clustering analysis. The method starts from an incidence matrix that represents the relationship of each part and machine. Although this clustering matrix method is difficult to represent and visualize when the incidence matrix gets large, some of the work is useful to this research. The transformation of the incidence matrix into a similarity measure is defined as:

$$p_{ij} = \sum_{k=1}^m \delta(a_{ik}, a_{jk}), \text{ where } \delta(a_{ik}, a_{jk}) = \begin{cases} 1, & \text{if } a_{ik} = a_{jk} \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

and an integer programming (IP) model to maximize the total sum of similarity is formulated as

$$\text{Maximize} \quad \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_{ij} \quad (4.2)$$

$$\text{Subject to} \quad \sum_{j=1}^n x_{ij} = 1, \text{ for all } i = 1, \dots, n \quad (4.3)$$

$$\sum_{j=1}^n x_{jj} = w, \text{ where } w \text{ is number of part families} \quad (4.4)$$

$$x_{ij} \leq x_{jj}, \text{ for all } i = 1, \dots, n \text{ and } j = 1, \dots, n \quad (4.5)$$

$$x_{ij} \in \{0, 1\} \quad \text{for all } i, j \quad (4.6)$$

Constraint (4.3) ensures that each part belongs to exactly one part family, (4.4) specifies the required number of part families, (4.5) ensures part  $i$  belongs to part family  $j$  only when this part family is formed, and (4.6) ensures integrality of the decision variables. Solving the above IP results in part families and the corresponding machine cells. Although the constraints are not suitable for the assignment problem, the similarity matrix combined with the IP model may be useful.

The assignment problem belongs to pure IP class of problems since all decision variables are restricted to 0-1 values. The assignment problem has been extensively studied for different applications, however, most of them focus on asymmetric assignment problems. The symmetric problems require that all assignments should be distinct to each other, which the asymmetric problems don't. An IP model for symmetric assignment, more specifically a Balanced Symmetric Assignment (BSA) problem, with a minimization objective function is formulated in [58]; a branch and bound method is proposed to solve it.

A pure 0-1 IP formulation for double feeder assignment problem with maximization of similarity values is:

$$\text{Maximize} \quad \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_{ij} \quad (4.7)$$

$$\text{Subject to} \quad \sum_{i=1}^n x_{ij} = 1, \quad \text{for } j = 1 \text{ to } n \quad (4.8)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad \text{for } i = 1 \text{ to } n \quad (4.9)$$

$$x_{ij} \in \{0, 1\} \quad \text{for all } i, j \quad (4.10)$$

$$x_{ij} = x_{ji} \quad \text{for all } i, j \quad (4.11)$$

where

$$x_{ij} = \begin{cases} 1 & \text{if component } i \text{ and } j \text{ are assigned to the same feeder} \\ 0 & \text{otherwise} \end{cases}$$

The constraint (4.11) forces the assignment to be symmetric, that is, each pair is distinct. Since the assignment problem is a general 0-1 integer-programming problem, it is a NP-complete class problem [30,40]. This suggests that the difficulty of solving the problem will grow as the problem size increases. Therefore, the optimal solution of the double feeder assignment problem depends upon the number of components involved. Tradeoff between solution time and quality of results is also a factor to be considered in selecting an appropriate solution technique. The two problems are actually related because exact algorithms usually require a large amount of CPU memory as computations progress. This further



slows down the computation process. In most cases, the solution progress is forced to abort either because the CPU memory is exhausted or the required computation time is unacceptable. Heuristic algorithms that solve for near-optimal solutions are commonly developed to increase efficiency and reduce computer resource dependency.

## 4.2 Computation Modeling

The double-feeder assignment IP model, as formulated in (4.7) to (4.11), states that  $n$  components are assigned into  $n/2$  double feeders. It is extended to a triple-feeder assignment and a generalized multi-track feeder assignment model.

### 4.2.1 Multi-track Feeder Assignment Model – Generalized IP Models

**Triple-feeder Assignment model:** The objective function remains the same – to maximize the total sum of similarity value of the assignment. The constraints are that each assignment must be distinct from the others, and each component can only be assigned once.

$$\text{Maximize} \quad \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n p_{ijk} x_{ijk} \quad (4.12)$$

$$\text{Subject to} \quad \sum_{j=1}^n \sum_{k=1}^n x_{ijk} = 1, \quad \text{for } i = 1 \text{ to } n \quad (4.13)$$

$$\sum_{i=1}^n \sum_{k=1}^n x_{ijk} = 1, \quad \text{for } j = 1 \text{ to } n \quad (4.14)$$

$$\sum_{i=1}^n \sum_{j=1}^n x_{ijk} = 1, \quad \text{for } k = 1 \text{ to } n \quad (4.15)$$

$$x_{ijk} \in \{0, 1\}, \quad \text{for all } i, j, k \quad (4.16)$$

$$x_{ijk} = x_{ikj} = x_{jik} = x_{jki} = x_{kij} = x_{kji} \quad \text{for all } i, j, k \quad (4.17)$$

where

$$x_{ijk} = \begin{cases} 1 & \text{if components } i, j, k \text{ are assigned to the same feeder} \\ 0 & \text{otherwise} \end{cases}$$

The symmetric matrix is expanded to three dimensions and the symmetry property still holds.

Now consider that  $n$  types of components are to be assigned to  $r$ -track feeders. Assume that there are  $m$  feeders and  $n = m \times r$ . **The Multi-track feeder Assignment model** is formulated as:

$$\text{Maximize} \quad \sum_{c_1=1}^n \sum_{c_2=1}^n \dots \sum_{c_r=1}^n p_{c_1 c_2 \dots c_r} x_{c_1 c_2 \dots c_r} \quad (4.18)$$

$$\text{Subject to} \quad (1) \sum_{c_2=1}^n \sum_{c_3=1}^n \dots \sum_{c_r=1}^n x_{c_1 c_2 \dots c_r} = 1, \quad \text{for } c_1 = 1 \text{ to } n \quad (4.19)$$

$$(2) \sum_{c_1=1}^n \sum_{c_3=1}^n \dots \sum_{c_r=1}^n x_{c_1 c_2 \dots c_r} = 1, \quad \text{for } c_2 = 1 \text{ to } n \quad (4.20)$$

$$\vdots$$

$$(r) \sum_{c_1=1}^n \sum_{c_2=1}^n \dots \sum_{c_{r-1}=1}^n x_{c_1 c_2 \dots c_r} = 1, \quad \text{for } c_r = 1 \text{ to } n \quad (4.21)$$

$$x_{c_1 c_2 \dots c_r} \in \{0, 1\} \quad \text{for all } c_1, c_2, \dots, c_r \quad (4.22)$$

$$x_{\{c_1 c_2 \dots c_r\}} \text{ are of the same value for all permutations of } \{c_1, c_2, \dots, c_r\} \quad (4.23)$$

where

$$x_{c_1 c_2 \dots c_r} = \begin{cases} 1 & \text{if components } c_1, c_2, \dots, c_r \text{ are assigned together} \\ 0 & \text{otherwise} \end{cases}$$

#### 4.2.2 Multi-track Feeder Assignment Model – Application Models

The difficulties in applying the above generalized formulation to large-scale problems result from too many decision variables and constraints. By taking the advantage of the symmetry property, the amount of decision variables and constraints can be significantly reduced. For example, the **Simplified Double-feeder Assignment (SDA)** model can be formulated as:

$$\text{Maximize} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^n p_{ij} x_{ij} \quad (4.24)$$

$$\text{Subject to} \quad \sum_{\substack{j=1 \\ j \neq i}}^n x_{(ij)_{\text{ordered}}} = 1, \quad \text{for } i = 1 \text{ to } n \quad (4.25)$$

$$\text{where } x_{(ij)_{\text{ordered}}} = \begin{cases} x_{ji} & \text{if } i > j \\ x_{ij} & \text{if } i < j \end{cases}$$

$$x_{ij} \in \{0, 1\}, \quad \text{for all } i, j \quad (4.26)$$

Notice that only the upper or lower half of the matrix (i.e., above or below the diagonal) is used as decision variables. The number is reduced from  $n^2$  to  $n(n-1)/2$ . The constraints are reduced from  $2n$  to  $n$  due to the symmetry property.

The **Simplified Triple-feeder Assignment (STA)** model is:

$$\text{Maximize} \quad \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^n p_{ijk} x_{ijk} \quad (4.27)$$

$$\text{Subject to} \quad \sum_{j=1}^n \sum_{k=j+1}^n x_{(ijk)_{\text{ordered}}} = 1, \quad \text{for } i = 1 \text{ to } n, j \neq i, k \neq i \quad (4.28)$$

$$x_{ijk} \in \{0, 1\}, \quad \text{for all } i, j, k \quad (4.29)$$

where

$$x_{ijk} = \begin{cases} 1 & \text{if component } i, j, k \text{ are assigned together} \\ 0 & \text{otherwise} \end{cases}$$

Similarly, the **Simplified Multi-track feeder Assignment (SMA)** model is:

$$\text{Maximize} \quad \sum_{c_1=1}^{n-r+1} \sum_{c_2=c_1+1}^{n-r+2} \dots \sum_{c_r=c_{r-1}+1}^n p_{c_1 c_2 \dots c_r} x_{c_1 c_2 \dots c_r} \quad (4.30)$$

$$\text{Subject to} \quad \sum_{c_2=1}^n \sum_{c_3=c_2+1}^n \dots \sum_{c_r=c_{r-1}+1}^n x_{(c_1 c_2 \dots c_r)_{\text{ordered}}} = 1, \quad \text{for } c_1 = 1 \text{ to } n, c_2, \dots, c_r \neq c_1 \quad (4.31)$$

$$x_{c_1 c_2 \dots c_r} \in \{0, 1\}, \quad \text{for all } c_1, c_2, \dots, c_r \quad (4.32)$$

where

$$x_{c_1 c_2 \dots c_r} = \begin{cases} 1 & \text{if components } c_1, c_2, \dots, c_r \text{ are assigned together} \\ 0 & \text{otherwise} \end{cases}$$

These simplified models are more applicable in practice because of fewer decision variables and constraints, thus requiring less computer resources and eliminating unnecessary search for optimum. The comparison of general and simplified models is summarized in Table 4.4.

Table 4.4 Comparison of general and simplified assignment models

Assignment IP Model	Number of decision variables	Number of constraints
General Double feeder	$n^2$	$2n$
Simplified Double feeder	$C_2^n = n(n-1)/2$	$n$
General Triple feeder	$n^3$	$3n$
Simplified Triple feeder	$C_3^n = n(n-1)(n-2)/6$	$n$
General Multi-track feeder	$n^r$	$m$
Simplified Multi-track feeder	$C_r^n = \frac{n!}{(n-r)!r!}$	$n$

$n$ : number of component types;  $r$ : number of feeder tracks

#### 4.2.3 Implementation of the IP Model

Commercial mixed integer programming solvers are typically limited by the number of decision variables and constraints. Also, each solver implements different algorithms (simplex method, interior-point method, branch-and-bound, etc.) and offers different options. Solvers also differ in terms of speed, accuracy, and number of iterations that can be performed.

AMPL with CPLEX solver [29] was chosen to solve the assignment problem. The reason for selecting AMPL is that its model and data can be constructed

separately. For new problems, only data file needs to be constructed while the model file remains unchanged. CPLEX is the optional solver that works with AMPL for mixed integer programming. CPLEX is designed to solve large, complex problems where other linear programming solvers fail or are unacceptably slow.

As an example, a small size problem with six boards and nine components is presented here to illustrate the double-feeder and triple-feeder assignment models. The board-component incidence matrix  $A_{6 \times 9}$  is

$$A_{6 \times 9} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{matrix} \quad (4.33)$$

For the double-feeder case, the component-to-component similarity values are calculated by using (4.1), except that

$$\delta(a_{ik}, a_{jk}) = \begin{cases} 1, & \text{if } a_{ik} = a_{jk} = 1 \\ 0, & \text{otherwise} \end{cases}$$

The symmetric similarity matrix  $P_{9 \times 9}$  is:

$$P_{9 \times 9} = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix} & \begin{bmatrix} - & 2 & 0 & 2 & 2 & 2 & 1 & 2 & 2 \\ & - & 1 & 4 & 3 & 2 & 3 & 3 & 3 \\ & & - & 1 & 0 & 0 & 1 & 0 & 0 \\ & & & - & 2 & 2 & 4 & 2 & 2 \\ & & & & - & 1 & 1 & 2 & 2 \\ & & & & & - & 2 & 2 & 2 \\ & & & & & & - & 1 & 1 \\ & & & & & & & - & 3 \\ & & & & & & & & - \end{bmatrix} \end{matrix} \quad (4.34)$$

The AMPL model is given in Appendix A.1. The data file for (4.34) is listed in Appendix A.2. The result shows that one of the optimal assignments is (1,6) (2,5) (4,7) (8,9), leaving component 3 alone. The optimal objective value is 12. Notice that a dummy component number is added in the data file to pair with the last unassigned component.

The similarity matrix for the triple-feeder case is constructed similar to the double feeder formulation with modification on (4.1)

$$p_{ijk} = \sum_{l=1}^m \delta(a_{il}, a_{jl}, a_{kl}), \text{ where } \delta(a_{il}, a_{jl}, a_{kl}) = \begin{cases} 1, & \text{if } a_{il} = a_{jl} = a_{kl} = 1 \\ 0, & \text{otherwise} \end{cases} \quad (4.35)$$

This three-dimensional similarity matrix is listed in Appendix A.3. The AMPL model and the data file are listed in Appendices A.4 and A.5. One of the optimal assignments is (1,3,5) (2,8,9) (4,6,7). The optimal objective value is 5.

The construction of similarity matrix for higher dimensions is similar to triple feeder case. However, the representation of matrix is more complicated as the problem dimension grows.

#### 4.2.4 Discussion

The use of an IP solver is strongly recommended as long as the problem size allows for its use. This is because the resulting solution is optimal. Limitations to the use of IP solvers increase as problem size increases. Issues include:

1. **Complex model and data construction:** As the problem dimension increases, i.e., the number of feeder tracks increases, the construction of the IP model becomes more complicated. Moreover, the data file also becomes difficult to handle as the problem size grows.

2. **Hardware limitations:** Without enough computer hardware resources, the computation time may be longer than would be acceptable.
3. **Software limitations:** Even though the computer resources may be unlimited, software limitations is another issue. Commercial solvers have limitations on the number of constraints and decision variables and on the maximum number of branches or nodes that can be searched.
4. **Constructions of similarity matrix:** The real challenge is the construction of the multiple-dimension similarity matrices. It is not difficult to realize that such transformation will be a time consuming and prone to error.

It is observed that a poor lower-dimension assignment will never produce a good higher-dimension assignment. For example, if  $p_{ij}$  is zero, then all combinations of  $p_{ij}$ ,  $p_{ji}$ ,  $p_{i\cdot}$ ,  $p_{\cdot j}$ ,  $p_{\cdot i}$ , and  $p_{i\cdot}$  in a triple feeder similarity matrix will be zeros. With this property, a partial similarity matrix, generally the non-zeros, is sufficient to solve the problem. It reduces the load of constructing a complete but huge similarity matrix, and eliminates a lot of unnecessary searches. It also provides a ground to solve for higher dimension problems.

#### 4.3 Heuristic Development

Multi-track systems require more slots than double feeders do. Therefore, the system does not benefit from multi-track feeders to increase its capacity as much as with double feeders. The heuristic development will focus on double feeders. The results are then extended to multiple-feeder problems.

The first heuristic, Min-Max (MMX), is developed to solve all types of multi-track feeder assignment problems. The components with non-zeros similarity values are first assigned. The unassigned components, whose similarity values are zeros, are then randomly grouped since they do not help to improve the assignment.

The second heuristic, **SWAP**, starts from an initial assignment and performs neighborhood search for local optimum. Components are reassigned if the local optimal assignment is different from the initial assignment.

The third heuristic, **INTEG**, integrates commercial software, MMX and SWAP with a problem size reduction procedure in a way that takes all advantage of each. The size reduction procedure is designed for the double feeder case.

These three heuristics are briefly described below.

#### **4.3.1 Min-Max (MMX) Heuristic – A Greedy Search Heuristic Algorithm**

This heuristic uses greedy search to approach the symmetric assignment for all types of multi-track feeder problems. The heuristic procedure for the double feeder case is described as below; the pseudo-code is provided in Appendix A.6.

1. Construct a non-zero assignment set  $S = \{x_{(1)}, x_{(2)}, \dots, x_{(n)}\}$ , whose similarity values are  $\{p_{(1)}, p_{(2)}, \dots, p_{(n)}\} = \{p_{i_1 j_1}, p_{i_2 j_2}, \dots, p_{i_n j_n}\}$ , sorted in descending order with ties broken randomly.
2. Enter the top un-crossed assignment with the largest similarity value, say  $x_{ij}$ , to the solution set.
3. Cross out  $x_{i\bullet}$  and  $x_{\bullet j}$ .
4. Repeat steps 2, 3 until all assignment variables in  $S$  are crossed out.
5. Randomly assign the rest of the unassigned components.

This heuristic selects the assignment with the largest non-zero similarity value to enter the solution set, while at the same time eliminating the remaining assignments associated with the entering components. After all the members in the non-zero assignment set are crossed out, the remaining unassigned components are all mutually exclusive to each other with zero similarity values. A random and



symmetric selection assignment is performed to complete the multi-track feeder assignment.

As the number of feeder tracks gets larger, the probability that a group of components has a zero similarity value increases. In other words, the ratio of zeros in a four-track feeder similarity matrix will likely be higher than in a double-feeder system. For this reason, this heuristic could be more efficient for multi-track problems because of less component combinations with non-zero similarity values.

The disadvantage is the relatively low solution quality because the search process does not cover enough solution space. The entering and crossing out steps may eliminate a better assignment combination. Repeated searching with different initial assignments, typically from one of the previous bests, can compensate for this shortcoming. The final assignment is chosen from the results. However, a tradeoff between the search time and solution quality must be made.

#### 4.3.2 SWAP – A Neighborhood Search Heuristic Algorithm

The SWAP heuristic performs neighborhood search and exchanges components between two assignments if a better assignment is found. The procedure, based on double feeder configuration, is described as below and the pseudo-code is provided in Appendix A.7.

- Step 1. Define  $v = n/2$ , and construct an initial symmetric solution set  $S = \{x_{(1)}, x_{(2)}, \dots, x_{(v)}\}$  whose corresponding similarity values are  $\{p_{(1)}, p_{(2)}, \dots, p_{(v)}\} = \{p_{i_1 j_1}, p_{i_2 j_2}, \dots, p_{i_v j_v}\}$ .
- Step 2. Compare two assignments, say  $x_{(1)}$  and  $x_{(2)}$ , with components  $i_1, j_1, i_2$ , and  $j_2$ . The three possible combinations are:  $(i_1, j_1)-(i_2, j_2)$ ,  $(i_1, j_2)-(i_2, j_1)$ , and  $(i_1, i_2)-(j_1, j_2)$ .

- Step 3. Perform a neighborhood search for each combination. If a better assignment is found, reassign the components.
- Step 4. Repeat steps 2 and 3 until comparisons between all assignments are complete.

Two key factors affect this heuristic efficiency and quality: the number of combinations between two assignments and the initial solution set  $S$ . This heuristic compares all possible combinations between two assignments to determine the local optimum. For a double-feeder case the possible combinations between two assignments is  $C_2^4/2$ , which is 3; and for triple-feeder case it is  $C_3^6/2$ , or 10. Although the combinations are increased from 3 to 10, the number of assignments to be compared are decreased from  $n/2$  to  $n/3$ . Since this heuristic is designed to improve the solution based on the initial assignment, the quality of the initial assignment affects the final assignment. For this reason, three initial assignments are chosen to initiate SWAP heuristic:

1. **MMX** assignment: The result from Min-Max heuristic are used as a starting point for the SWAP procedure.
2. Usage-based assignment (**USG**): Components with the same or close usage frequency are assigned together. The component usage frequency is the sum of the columns of PCB incidence matrix. This type of assignment was first used in industry to deal with the double feeder assignment.
3. Random feeder assignment (**RFA**): Components are randomly selected for assignment. This is thought to be a no-plan assignment.

#### 4.3.3 *Integrated Heuristic with Problem Size Reduction (Double feeder case)*

As discussed previously, the assignment problem has been extensively studied in literature. There are several optimal algorithms to deal with the asymmetric assignment problems. One of them is the Hungarian Algorithm [42], which

can optimally solve balanced asymmetric assignment problems by utilizing the primal-dual method. The procedure for this algorithm is:

**Step 1. Matrix reduction:** In a  $n \times n$  square matrix,

- i. For each row, subtract the row minimum from each row.
- ii. For each column with all positive entries, subtract the column minimum from each column.

**Step 2. Find the initial allocation:** Define the admissible cell as the cell in the current reduced matrix with a value of zero. Allocations can be made among admissible cells only, and each row and each column can have at most one allocation. If more than one admissible cells are available, make an allocation arbitrarily. Cross out all other cells in the column or row of the allocated cell. The efficient way to select initial solution is:

- i. Identify the rows or columns with exactly one uncrossed zero first, or choose the rows or columns with the least number of uncrossed zeros. Break ties arbitrarily and select one uncrossed zero. If the zero is selected from a row, then cross out the column through the selected zero, or cross out the row through the zero if it is selected from a column.
- ii. Repeat Step 2i until all zeros are crossed out.
- iii. If exactly  $n$  lines have been crossed, then it represents an optimal solution. The zeros identified comprise a solution. If less than  $n$  lines have been drawn, an optimal solution is not yet found; go to Step 3.

**Step 3. Redistribute the zeros and go back to Step 2:**

- i. Subtract the minimum entry among the uncrossed cells from each uncrossed cell.

- ii. Add the minimum value obtained from Step 3i to each cell which is both horizontally and vertically crossed out.
- iii. Remove all lines and go to Step 2.

An example of Hungarian algorithm applied on (4.34) is given in Appendix A.8. For balanced symmetric assignment problems, the Hungarian algorithm does not guarantee optimal solutions. However the optimal solution from the Hungarian algorithm can be used as an upper or lower bound of two-dimensional symmetric assignment. Consider a solution set  $H$  for a double feeder assignment problem solved by Hungarian algorithm:

$$H = \{ x_{ij} = 1, \text{ for all } i, j \} \quad (4.36)$$

which is separated into two subsets

$$S = \{ x_{ij} = 1 \text{ and } x_{ji} = 1, \text{ for all } i, j \}, \text{ symmetric solution subset, and} \quad (4.37)$$

$$NS = \{ x_{ij} = 1 \text{ and } x_{ji} = 0, \text{ for all } i, j \}, \text{ asymmetric solution subset} \quad (4.38)$$

$$H = \{ S, NS \}$$

If subset  $NS = \Phi$ , then the solution  $H$  is optimum. If  $NS \neq \Phi$ , then the solution  $H$  is not symmetric and, consequently, not a complete solution. Sometimes, changing the similarity matrix column and rows' order will result in a different solution. It is also discovered that re-solving the sub-matrix formed by  $NS$  can result in more symmetric sub-solutions, and the optimal value can still be maintained. The reason is that if there are multiple optimum solutions, some have a symmetric subset whereas others do not. Changing the order of matrix rows and columns will have Hungarian algorithm pick different initial allocations and hence a different final solution. The Hungarian algorithm can thus be applied repeatedly to solve the asymmetric subset. The problem size reduction procedure is summarized below.

- Step 1. Construct a new symmetric matrix  $P_{NS}$  for all the elements in subset  $NS$ . Orders of matrix are rearranged.

- Step 2. Re-solve  $P_{NS}$  using the Hungarian algorithm.
- Step 3. Separate the solution into symmetric and asymmetric subsets,  $S$  and  $NS$ .
- Step 4. Enter symmetric subset  $S$  into solution set.
- Step 5. Repeat step 1 to 4 until one of the termination criteria is met: (a)  $NS = \Phi$ , (b) there is no further improvement, or (c) there is forced termination.

The question now is: Will the reduction procedure lose the optimality property of the Hungarian algorithm? The answer is: It does not. The proof of this property is as follows:

Let the first solution set obtained from the Hungarian algorithm be denoted as  $H = \{ S_1, NS_1 \}$ , where  $S_1$  and  $NS_1$  are sub-solution sets. The objective function value is

$$z_H(x) = z_{S_1}(x) + z_{NS_1}(x) \quad \text{for every assignment } x \quad (4.39)$$

It is known that the Hungarian algorithm solves for optimal solution, which means that  $z_H(x)$  is optimal objective value. Re-solve  $NS_1$  and obtain solution  $\{S_2, NS_2\}$ , and

$$z_{NS_1}(x) = z_{S_2}(x) + z_{NS_2}(x) \quad \text{for every assignment } x \quad (4.40)$$

Repeating the re-solving step for  $NS_{k-1}$ , the solution is  $\{ S_k, NS_k \}$ , and

$$z_{NS_{k-1}}(x) = z_{S_k}(x) + z_{NS_k}(x) \quad \text{for every assignment } x \quad (4.41)$$

Back substitution of these results yields

$$z_H(x) = z_{S_1}(x) + z_{S_2}(x) + \cdots + z_{S_k}(x) + z_{NS_k}(x) \quad \text{for } x \quad (4.42)$$

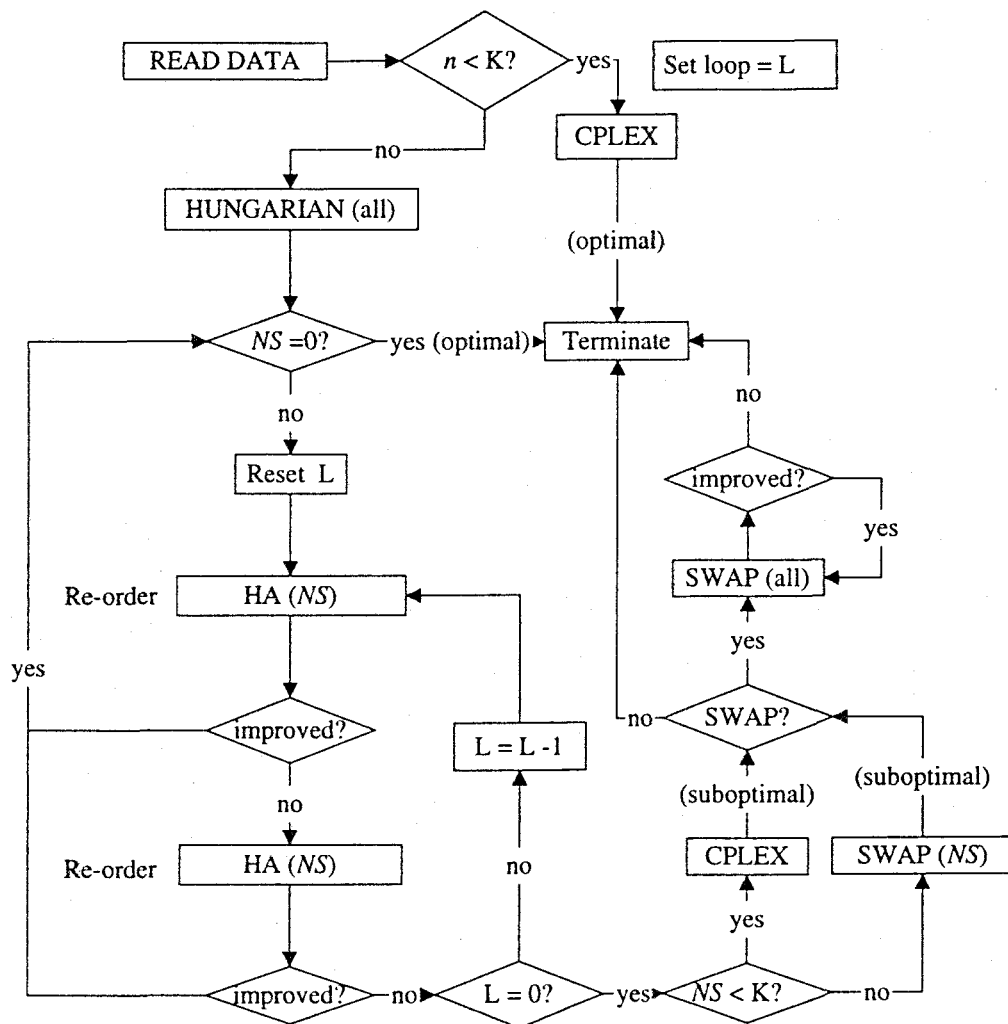
which still preserves the optimal objective value.

If  $NS$  cannot be completely solved, it cannot be proved that the optimal symmetric solution can be found in  $NS$ . In fact, the optimal solution obtained by Hungarian algorithm is convex [46,58,59], whereas the symmetric solution is a non-convex. This is the reason why the Hungarian algorithm can not be used or modified to solve symmetric assignment problems. Under some circumstances, a symmetric assignment can be proved optimal if:

1. It is a complete symmetric assignment obtained by the Hungarian algorithm, or
2. Its objective function value is the same as the Hungarian algorithm's, or
3. Its objective function value is less than the Hungarian algorithm's by 1, and the objective function value obtained by the Hungarian algorithm is an odd number.

Condition 2 above stands because the objective function value from the Hungarian algorithm can be used as the upper or lower bound of the symmetric assignment problem. Condition 3 holds because the objective function values of symmetric assignments are always even number, therefore the upper bound for an odd objective function value from the Hungarian algorithm will be less by 1. In other words, it is impossible to approach a complete symmetric assignment if the objective function value obtained by the Hungarian algorithm is an odd number.

The integrated heuristic for the double-feeder assignment problem will use the commercial package with IP models as long as the computation environment allows optimal solution to be obtained. The second choice is using the Hungarian algorithm. If a complete symmetric assignment can not be approached, the problem size will be reduced as much as possible. The SWAP or commercial package will solve the asymmetric subset if allowed. The heuristic flow is displayed in Figure 4.1.



K: the maximum size allowed to run on CPLEX  
 L: the number of loops to perform size reduction

Figure 4.1 Integrated Heuristic Flow and Setup Options

## 4.4 Evaluation Study

Three sets of test problems are used to verify and investigate the performance of the heuristics developed for the double-feeder assignment problem. These data sets are obtained from literature, through use of a random number generator, and from industry. A brief description of the data sets follows.

### 4.4.1 Description of Test Problems

The first set (Data set 1) contains 8 mid-sized problems. They are taken from the OR-Library and published in [8,9], and were used in the asymmetric assignment problem. The problem size ranges from 100×100 to 800×800, and the optimal solution values for the problems are known. This problem set is used to verify the Hungarian algorithm. A summary of Data set 1 is given in Table 4.5.

Table 4.5 Data Set 1 from OR-Library

Data set	L1	L2	L3	L4	L5	L6	L7	L8
Size*	100	200	300	400	500	600	700	800
Minimum**	1	1	1	1	1	1	1	1
Mean**	50.96	50.97	51.05	51.03	51.06	51.01	51.01	50.98
Maximum**	100	100	100	100	100	100	100	100
Std. Dev.**	28.50	28.56	28.57	28.57	28.59	28.6	28.58	28.58
Opt. Value	305	475	626	804	991	1176	1362	1552

\* square matrix

\*\* statistics of the matrix elements

The second set (Data set 2) contains five subsets of matrices ranging from 18×18 to 2000×2000 with different statistical distributions. They are created using a random number generator designed for construction of double feeder similarity matrix because it is impossible to control the similarity matrix distributions from a



board-component incidence matrix. The generator produces the similarity matrix by controlling:

1. the number of components and PCBs involved,
2. the random seed number,
3. the minimum and maximum number in the matrix, and
4. the mean and standard deviation of the matrix.

The configuration of these test problems is summarized below:

Table 4.6 Data Set 2 from the random number generator

Data set	R1	R2	R3	R4	R5
# of PCBs	1000	1000	1000	1000	1000
Seed	random	random	random	random	random
Minimum	0	0	0	0	0
Mean	50	100	150	200	250
Maximum	100	200	300	400	500
Std. Dev.	50	100	150	200	250
# of Components for each set: 18, 50, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000					

Data set 2 covers a range of problem sizes. The 18×18 problems are tested on AMPL and CPLEX. The other problems are tested for each heuristics except SWAP with USG initial assignment. The configuration of the problem set also explores the effects of distribution of the matrix and relative size of components and PCBs on performance of heuristics.

The third problem set (Data set 3) is obtained from Tektronix Inc., Beaverton, Oregon. These are transformed from seven original PCB-component incidence matrices. They are solved using each heuristics and are later used in the sequencing problems. Table 4.7 summarizes their characteristics.

Table 4.7 Data Set 3 from industry

	DataSet						
	A	B	C	D	E	F	G
# of PCBs	437	458	281	744	620	608	843
# of Components	695	625	547	796	769	712	826
Minimum	0	0	0	0	0	0	0
Mean	1.40	1.68	1.55	1.72	1.68	1.79	1.84
Maximum	185	185	127	287	268	240	329
Std. Dev.	6.00	7.20	6.01	8.22	8.07	8.22	9.11

#### 4.4.2 Computational Results and Analysis

The mid-sized problems with known optimal solutions, were obtained from the literature, are applied on the Hungarian algorithm. An efficient program for the Hungarian algorithm was coded by Carpaneto and Toth [15] in Fortran and tested on problem size ranging from 50 to 200. The program is adapted and improved with study by Wright [79], and finally re-coded in Visual Basic 5.0. Note that the optimal solution values given by the Hungarian algorithm are solved as minimization problems. Therefore, the sign of the similarity values in the original data is changed because the double feeder assignment problem is to be solved as maximization problem. For large-sized problems the optimal solutions are unknown, except those that meet the optimal criteria in section 4.3.3. The other heuristic algorithms are also coded in Visual Basic and tested on a Dell 300MHz with 64 MB RAM in Microsoft Windows NT 4.0 operation system.

For each data set, the optimal solution value solved by the Hungarian algorithm for asymmetric assignment problem (i.e., upper bound for symmetric assignment problem) is compared with the results from the other heuristics. The average solution times (in CPU seconds) are also reported. The results for Data set 1 are shown in Table 4.8.

Table 4.8 Computational results for the Data set 1

Problem	Known Optimal value	10-runs										Diff. $\sigma$ %	Average CPU secs	CPU time std.dev.
		1	2	3	4	5	6	7	8	9	10			
L1	305	*	*	*	*	*	*	*	*	*	*	0	0.19	0.01
L2	475	*	*	*	*	*	*	*	*	*	*	0	0.50	0.01
L3	626	*	*	*	*	*	*	*	*	*	*	0	0.89	0.01
L4	804	*	*	*	*	*	*	*	*	*	*	0	1.48	0.03
L5	991	*	*	*	*	*	*	*	*	*	*	0	2.13	0.02
L6	1176	*	*	*	*	*	*	*	*	*	*	0	2.79	0.02
L7	1362	*	*	*	*	*	*	*	*	*	*	0	4.08	0.02
L8	1552	*	*	*	*	*	*	*	*	*	*	0	5.25	0.06

\* : the same as optimal value  
Each run solves rearranged matrix

The results show that the Hungarian algorithm finds the optimal solution in all 10 test runs. Each run solves the rearranged matrix. The average execution times are all within five CPU seconds.

Since Data set 2 is produced by random number generator, there is no associated incidence matrix. All heuristics are used except SWAP/USG. Each data set is re-arranged and repeatedly solved by the Integrated heuristic for five runs to search for the best solution. The results are shown in Appendix A.9 to A.13 with the optimal solutions boxed. It shows that the Hungarian algorithm optimally solved the symmetric assignment for all 18×18 problems, and up to the 200×200 problem. The Integrated heuristic dominates the solution quality for all data sets. The computation times are within 100 seconds. Figure 4.2a displays the number of best solutions found by each heuristics. Figure 4.2b shows those solutions that also can be verified as optimal solutions.

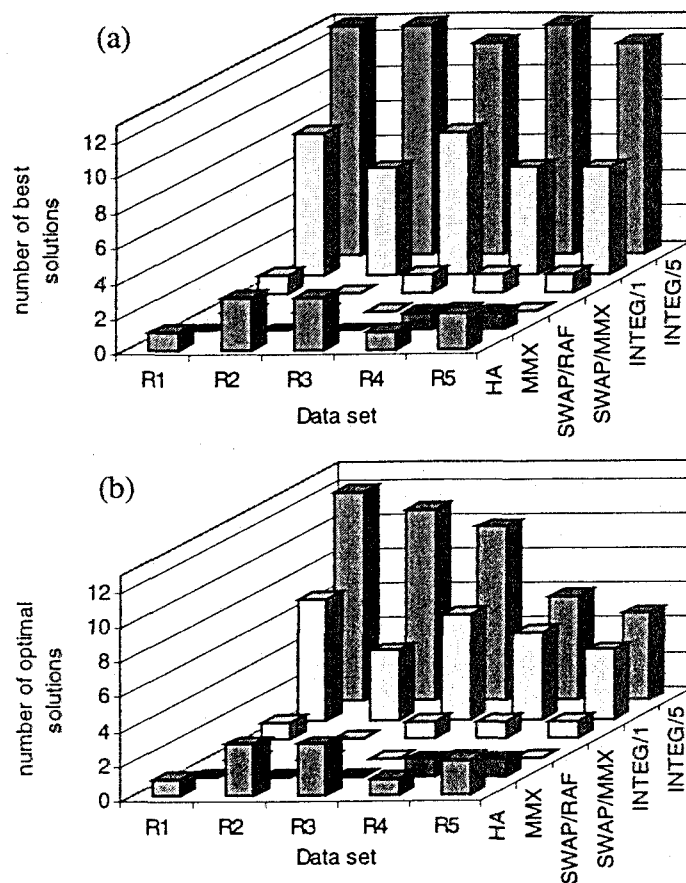


Figure 4.2 The number of (a) best solutions, and (b) optimal solutions found by each heuristics.

Figure 4.2a shows that most of best solutions are found by the Integrated heuristic with five runs. In Figure 4.2b, most of the optimal solutions are also found by the Integrated heuristic, yet they are more difficult to be obtained as the spread of the data increases. Figure 4.3 displays the relationships among solution time, data size, data distribution, and heuristics. Note that the spread of the data sets increases from R1 to R5 (see Table 4.6).

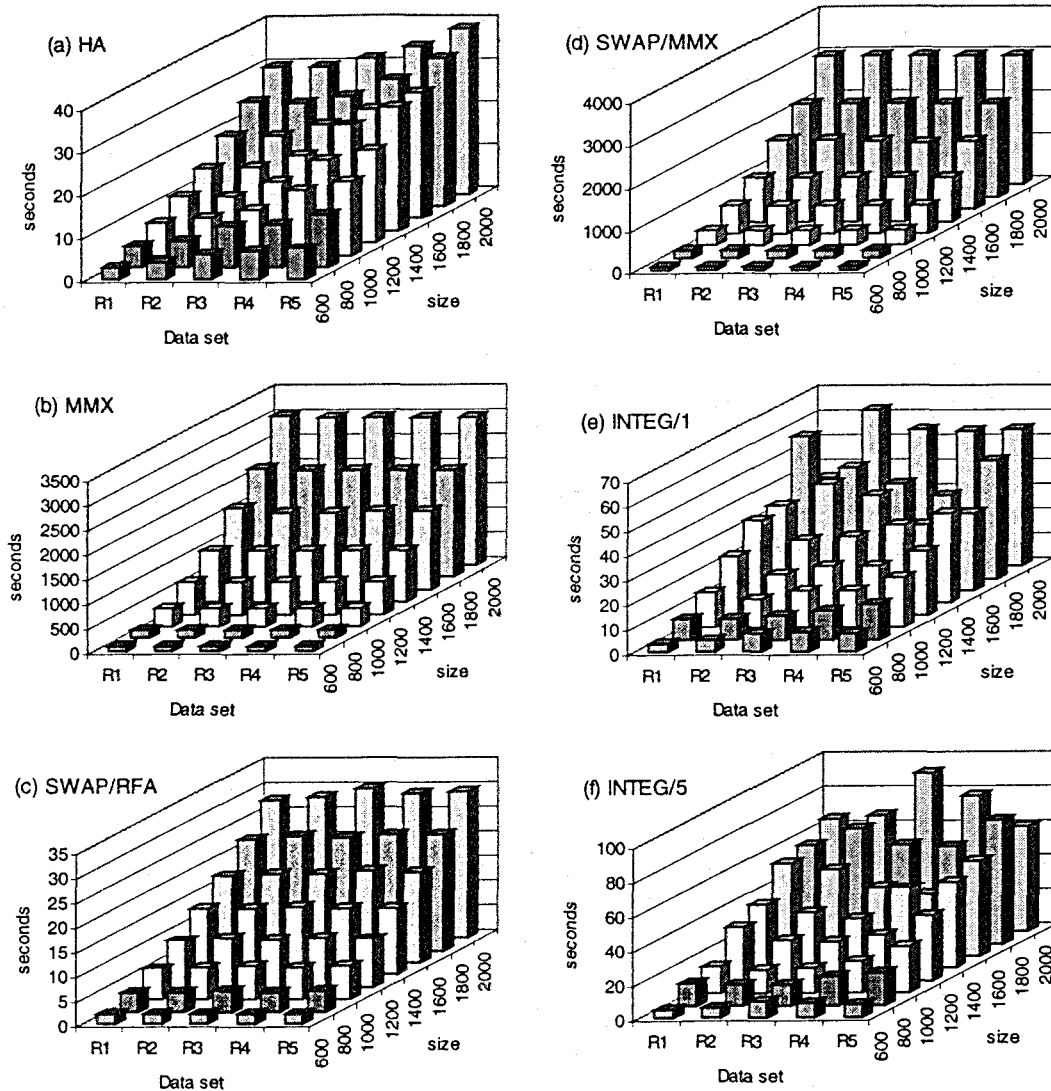


Figure 4.3 Dependency of computation time on data size, distribution, and heuristics.

When the problem size is larger than 1000, the MMX becomes unacceptably slow (Figure 4.3b). Since SWAP/MMX (Figure 4.3d) requires MMX to construct initial solution, most of the computation time (over 97%) is spent on MMX. Although SWAP/RFA is the fastest heuristic, the solution quality is the poorest. These results are summarized below.

1. The Hungarian algorithm works well on small-sized problems, especially under 200×200. Figure 4.3a shows that it is the only one with a consistent pattern of computation time increasing with changes in data distribution and problem size.
2. The MMX is inefficient for large problems with many non-zero similarity values. The CPU time is dependent on problem size.
3. The SWAP heuristic is efficient for double feeder problems. The initial assignment construction uses most of the CPU time. The problem size is the another factor that affects its efficiency.
4. The Integrated heuristic has both high efficiency and solution quality for small and large size problems. The computation time is determined by the heuristic flow setup, data size and distribution.

The computation efficiency is further investigated by fitting the CPU times into several possible formats including linear, exponential, polynomial, powered, and logarithmic. The powered format, shown below, can explain more than 98% of data variation based on the R-squared values. The parameters and R-squared values for each data sets are listed in Table 4.9.

$$Y = \alpha \left( \frac{X}{200} \right)^\beta, \text{ where } Y = \text{CPU time in seconds, } X = \text{data size} \quad (4.43)$$

Table 4.9 The fitted functions for Data set 2

Heuristic	R1			R2			R3			R4			R5		
	$\alpha$	$\beta$	R <sup>2</sup>	$\alpha$	$\beta$	R <sup>2</sup>	$\alpha$	$\beta$	R <sup>2</sup>	$\alpha$	$\beta$	R <sup>2</sup>	$\alpha$	$\beta$	R <sup>2</sup>
HA	0.49	1.75	0.99	0.89	1.46	0.99	1.26	1.38	0.99	1.10	1.55	0.99	1.73	1.39	0.99
SWAP/RFA	0.20	2.16	0.99	0.21	2.14	0.99	0.21	2.15	0.99	0.22	2.13	0.99	0.20	2.18	0.99
INTEG	0.50	2.05	0.98	0.83	1.78	0.98	1.18	1.61	0.99	1.18	1.60	0.99	1.71	1.50	0.99
MMX	2.88	3.04	1.00	2.84	3.03	1.00	2.83	3.03	1.00	2.83	3.03	1.00	2.83	3.03	1.00

The relationships of data sets and parameters  $\alpha$  and  $\beta$  for each heuristics are displayed in Figure 4.4.

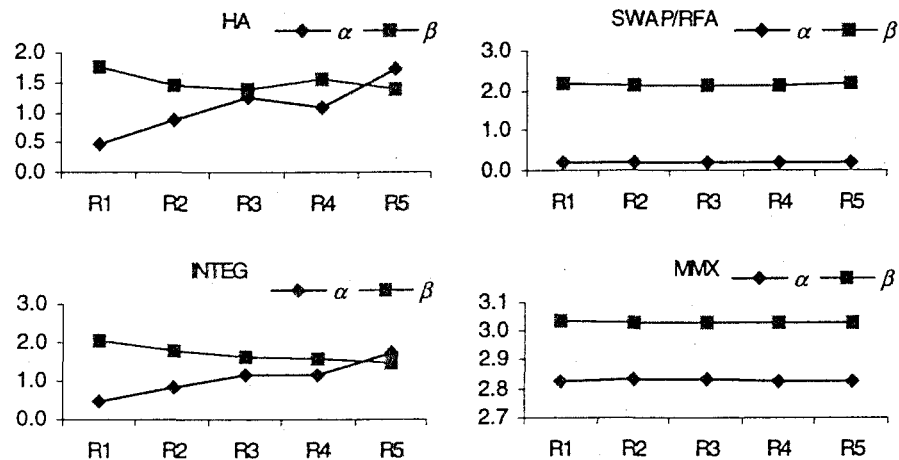


Figure 4.4 The relationships of data sets and fitted equation parameters  $\alpha$  and  $\beta$  in each heuristic.

Apparently, the fitted equations for SWAP/RFA and MMX are consistent in all data sets which means the independence of solution time to data distribution. In contrast, the equation parameters are significantly correlated to the data distribution in the Hungarian and the Integrated heuristics, confirming previous observations.

Data set 3 are tested for 10 runs on all heuristics including SWAP/USG. The Integrated heuristic is configured such that after two consecutive loops without improvement, the reduction procedures will end and then use SWAP/MMX to finish assignment. The CPLEX solver is used with priority if problem size can be reduced to its allowable limit. The results are shown in Appendix A.14. The boxed results represent the best solutions because the optimal values could not be verified.

Again the Integrated heuristic approach yields the greatest number of best solutions for all data sets, and the CPU times are all within 300 seconds. The similarity values from industry data are highly distributed compared to data sets 1 and 2. The data averages are skewed to the left with long-tailed shape. This major difference makes the Hungarian algorithm requires more search loops for initial allocations and for distinct admissible cells (zeros) in each column and row. Table 4.10 compares the actual solution times and the predicted values by using the fitted equation (4.43) with parameters  $\alpha$  and  $\beta$  from data set R5 in Table 4.9.

Table 4.10 Comparisons of actual and predicted computation time for industry data

Data set	Data size	HA	predicted	MMX	predicted	SWAP/RFA	predicted	INTEG	predicted
A	695	90.95	9.71	125.42	123.66	4.19	2.97	176.79	11.06
B	625	60.42	8.38	90.46	89.62	3.32	2.36	103.66	9.44
C	547	36.36	6.97	59.89	59.81	2.52	1.76	49.10	7.73
D	796	144.21	11.72	187.68	186.64	5.56	3.99	236.60	13.56
E	769	143.69	11.17	169.77	168.09	5.29	3.70	192.56	12.88
F	712	93.51	10.04	134.20	133.07	4.48	3.13	150.26	11.47
G	826	146.77	12.33	210.54	208.80	5.80	4.33	221.26	14.33

Two conclusions can be drawn from the above results:

1. The computation times by the MMX and SWAP/RFA heuristics are close to the predicted values. Thus, the efficiencies of these heuristics are mainly affected by data size.
2. The results from the Hungarian algorithm do not fit well with the equations at all. The confounding factor of data distribution has a higher effect than does data size. Since more than 50% of the computation time by the Integrated heuristic is spent on the reduction procedure, i.e., the Hungarian algorithm, the Integrated heuristic is also prone to be affected by data distribution.



Figure 4.5 displays the comparisons of the first reduction procedure between data sets 2 and 3 in terms of percentage of the solved symmetric assignments and similarity values.

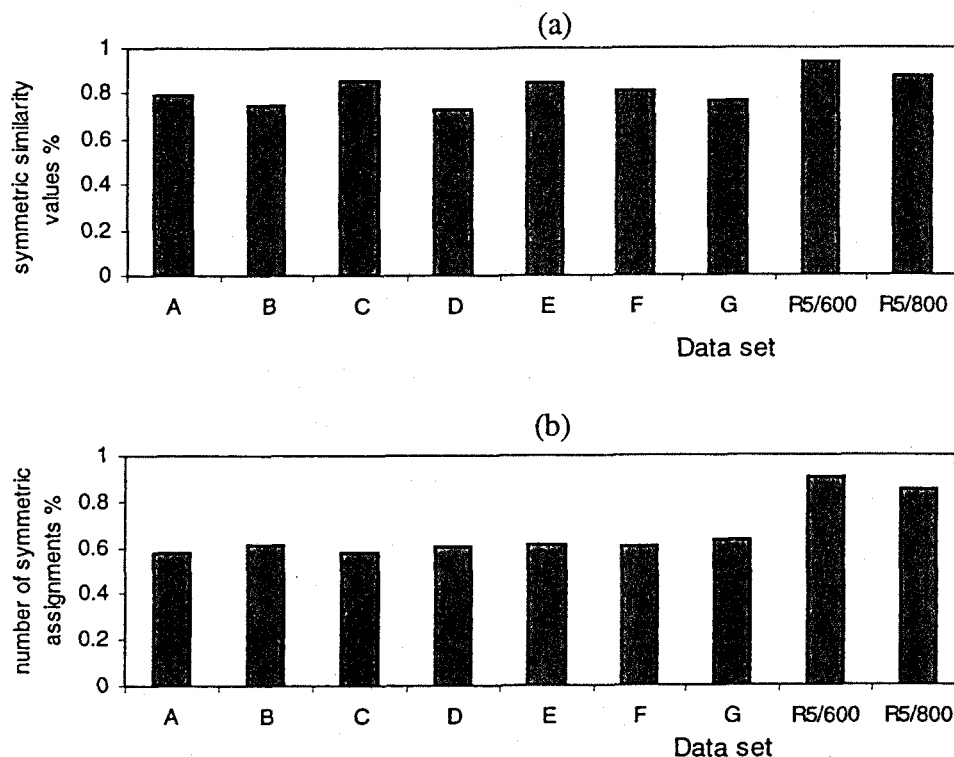


Figure 4.5 The percentage of a) symmetric similarity values, and b) number of symmetric assignments, solved by the first reduction procedure.

The figures show that the first reduction, or the Hungarian algorithm, solves about 90% of symmetric similarity values for Data set 2, and about 80% for Data set 3. However, only 60% of total assignments in industry data is reduced compared to over 85% reduction in simulated data. The low reduction rate requires more search loops before using SWAP heuristic or CPLEX solver to solve the assignments.

The SWAP heuristics with random and usage-based initial assignment require the least computation time to approach final solution. For large-sized

problems, they are more attractive than the Integrate or other heuristics, even though their solution qualities are not as good as the Integrated heuristics.

## 4.5 Chapter Summary and Conclusions

In this chapter, an integer programming model for multi-track feeder assignment is presented to solve the symmetric assignment problem. Heuristic algorithms are developed to solve large-scale problems due to the *NP*-complete property.

The MMX heuristic is constructed with selected and sorted search methods to deal with the multi-track problem. To improve the MMX efficiency and solution quality, the SWAP heuristic is developed with an initial symmetric assignment, to compare assignments and to search for local optimum. It proves to be high efficient if a high quality initial solution can be easily obtained. The Integrated heuristic is therefore designed to take advantage of the other methods to approach the best solution. Commercial solver is used whenever the problem size allows. The Hungarian algorithm is introduced to reduce the problem size by separating symmetric and asymmetric assignments. The SWAP heuristic with MMX initial solution then completes the assignment problem.

The heuristics are applied to three data sets to evaluate their performance. Data set 1 from the OR-Library with known optimal solution values is used to verify the coded programs and the efficiency of the Hungarian algorithm. It shows that both the algorithm efficiency and solution quality are quite good. Data set 2 is generated from random numbers with varied distribution parameters. The simulated data ranges from small to large problem sizes and has different spreads. Data set 3 is obtained from industry and used to evaluate the application of the developed heuristics on real-world problems.

The following conclusions can be drawn from the analysis.

1. The MMX heuristic is highly affected by data size and is not efficient for solving large-scale problems.
2. The Hungarian algorithm is efficient in reducing the problem size for large-scale symmetric assignment problems. However, the data distribution significantly affects the algorithm's performance.
3. The SWAP algorithm with an initial solution is the most efficient among all heuristics and is also independent of data distribution. The solution quality, however, is dependent on the quality of the initial assignment.
4. The Integrated heuristic finds most of the best solutions with impressive efficiency. Because of using the Hungarian algorithm to reduce problem size, its efficiency is also affected by the data distribution.
5. The industry data, because of their skewed distributions, require more solution time.

The effect of data distribution and the relationship of similarity value with the PCB assembly setup will be further investigated in chapter 6 where the assignment problem is examined in combination with board sequencing problem.

## CHAPTER 5    PCB SEQUENCING PROBLEM WITH MULTI-TRACK FEEDERS

This chapter discusses the PCB sequencing problem and its interaction with the multi-track feeder assignment problem. A component and board grouping methodology is developed to integrate with the component-to-feeder assignment method discussed in chapter 4. The sequencing procedure is based on the single feeder configuration. The performance is compared with past research due to a lack of studies related to double or multi-track feeders. The PCB sequence obtained from this methodology is further improved by an optimal tool switch plan called *Keep Tool Needed Soonest* or KTNS. This plan was designed to minimize the number of tool switches in flexible manufacturing systems. The tool switch is equivalent to the single feeder setup in this research. The KTNS plan is modified to work with component-to-feeder assignment and to apply to multi-track feeder problems.

### 5.1        Methodology

In the assignment problem, the component-to-component similarity matrix is used to assign similar components together. Also, a high similarity value usually reflects a group of PCBs that use the same components. This leads to the idea that if the PCBs can be grouped by similar components, then the feeder setup within the group can be minimized. In fact, in the PCB production environment, some boards require identical component types on the same machine but with different amounts or different placement positions. The layout and quantity of components will affect the processing times but not the feeder setup as long as the component types remain the same. Some small boards use components that are a subset of a large board. In this case, if the small boards can be grouped and sequenced, in any order, after the large board, then there will be no setup required. The above grouping methodology

can eliminate identical or possibly subset relationships between PCBs. The minimal setups can be approached by sequencing PCBs within and between groups. The methodology implemented in this research includes component grouping, PCB grouping, intra- and inter-group sequencing (IIGS), and feeder setup planning.

A number of studies and methods have been developed for the application of group technology in different production environments, as summarized in Chapter 3. Among these research, a tabular method developed by Sule [71] for machine grouping in cellular manufacturing, and used in component grouping and scheduling for sequencers [72], is adapted and modified to accomplish the grouping tasks in PCB sequencing. The PCB sequencing method IIGS is developed similar to [72], except the line-balancing problem is ignored.

A tool switch plan called *Keep Tools Needed Soonest* or KTNS developed by Tang and Denardo [75,76] deals with job scheduling problem for a flexible manufacturing machine. It has proved to be optimum for minimizing the number of tool switches for a specific job sequence. The KTNS policy is modified for solving the multi-track feeder problem and fine-tuning the PCB sequence obtained by IIGS procedure. The flowchart for the methodology is displayed in Figure 5.1.

The notations used in this chapter are as follows:

- $m$  total number of PCB types;
- $n$  total number of components;
- $r$  number of track of feeder;
- $A$   $(a_{ij})_{m \times n}$ , the PCB incidence matrix;
- $a_{ij}$   $ij$ th element of matrix  $A$ ;  $a_{ij} = 0$  or  $1$ ;
- $a_{i\bullet}$   $i$ th row of matrix  $A$ ; the required component types of PCB  $i$ ;
- $a_{\bullet j}$   $j$ th column of matrix  $A$ ; the usage of component  $j$  by PCBs;

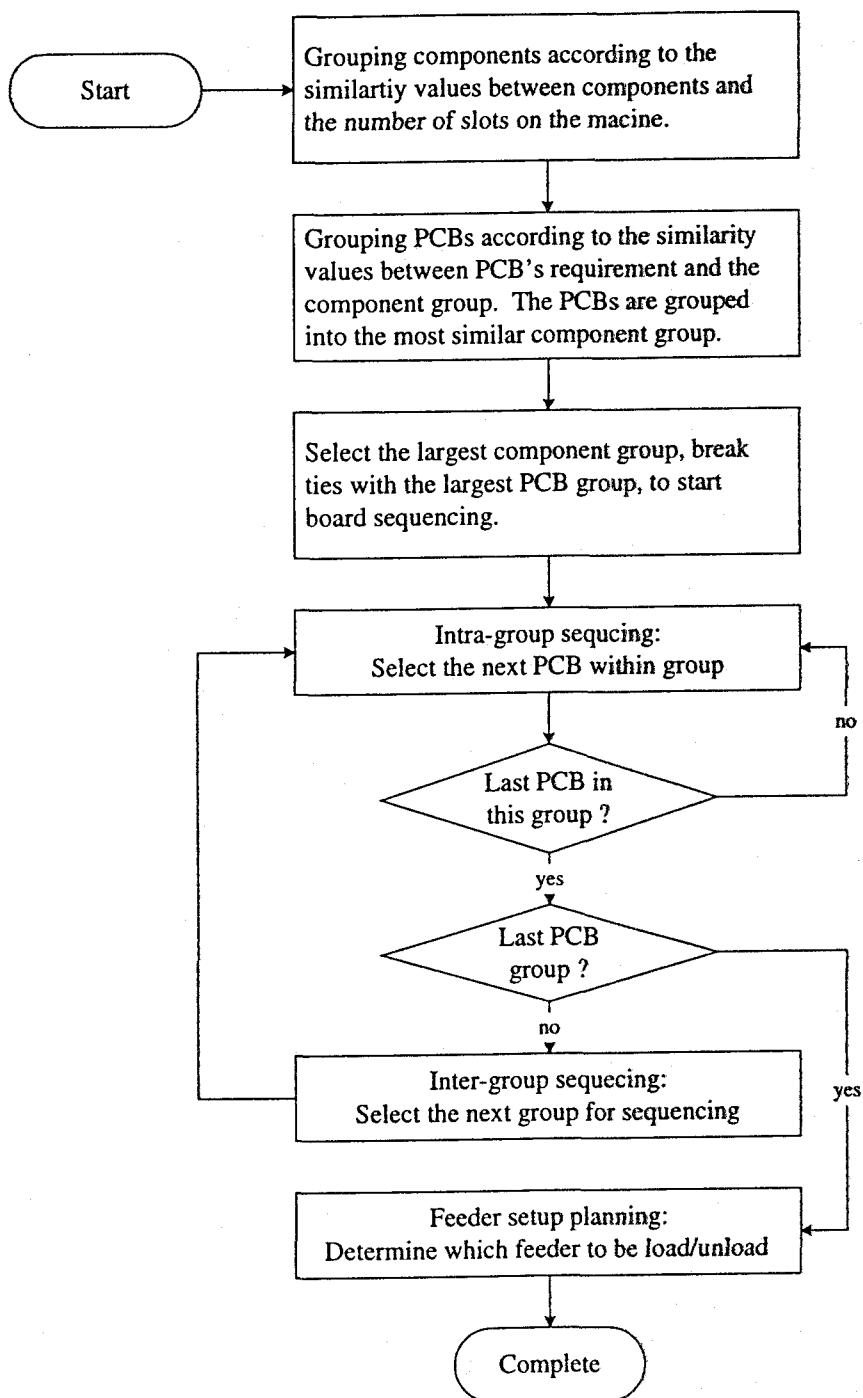


Figure 5.1 Methodology flowchart

- $P$   $(p_{ij})_{n \times n}$ , the component-component similarity matrix;  
 $p_{ij}$   $ij$ th element of matrix  $P$ , similarity value between component  $i, j$ ;  
 $p_i$  all similarity values associated with component  $i$ ;  
 $U$  threshold allowing a component to join a group,  $0 \leq U \leq 1$ ;  
 $G_w(k)$  the  $k$ th entered component in group  $w$ ;  
 $G_w[\bullet]$  number of components in group  $w$  at that instant;  
 $maxSV$  maximal similarity value in  $P$ ;  
 $LSV$  largest similarity value in  $P$  at that instant;  
 $CRA_{[w]}$  closeness ratio of component  $A$  with  $G_w = \sum_{k=1}^{G_w[\bullet]} p_{AG_w(k)} / G_w[\bullet]$ ;  
 $MCRB_{[wB]}$  maximal closeness ratio of component  $B$  with the entering group  $G_{wB}$ , in which  $G_{wB}[\bullet]$  is less than the number of slots;  
 $MTV$  minimal threshold value  $= LSV \times U$ ;  
 $maxS$  maximal similarity value between PCB's required components and all component groups;  
 $S_{ij}$  the similarity value between the required components of PCB  $i$  and component group  $j$  in PCB grouping procedures;  
 $EG$  the entering group number;  
 $E_1(k)$  current setup of component  $k$ , 1 if component  $k$  is on-line, 0 off-line, -1 on-line but not used by the remaining PCBs;  $n \geq k \geq 1$ ;  
 $E_2(k)$  planning setup of component  $k$  for next PCB, 1 if component  $k$  is on-line, 0 off-line, -1 on-line but not used by the remaining PCBs;  
 $KN$  number of components to be kept for the next PCB;  
 $sn$  number of slots a SMT machine is equipped;  
 $B_i$  The  $i$ th PCB in production sequence;  
 $RB_i$  number of component types required by  $B_i$ ;  
 $C(k)$  double assignment vector; component  $k$  is assigned with component  $C(k)$ ;  $n \geq k \geq 1$ ;  
 $or$  operator, for example  $1 \text{ or } 1 = 1$ ;  $1 \text{ or } 0 = 1$ ;  $0 \text{ or } 0 = 0$ .

## 5.2 Component and PCB Grouping

Using the tabular procedure developed by Sule [71,72], the components are grouped by their similarity values. The PCBs are then assigned to the closest group according to their requirements. There are several advantages of this tabular method, which this research can incorporate:

1. **Use of the component-to-component similarity matrix:** The most similar components are assigned to either an existing group or a new group according to the closeness ratio to each group and a user-defined threshold value. In the assignment problem, the components are also assigned to the multi-track feeders by their similarity values.
2. **User-defined group size:** The maximal group size and duplication of components in different groups are determined by users. An SMT machine has limited slot spaces for feeders, which is an implied restriction on the maximum number of components that can be installed on the machine at any instant. Therefore, it is necessary to control the number of components in a group to avoid exceeding the machine limits.
3. **Options for component duplication in different groups:** The threshold value mentioned above is also used in determining duplication of components in different groups. Originally, it is designed for grouping machines, in which duplication of machines in different groups involves cost and space consideration. Changing the threshold value will also result in different number of groups.

The grouping procedures are detailed in Appendix B.1. Figure 5.2 and 5.3 display the flowcharts for component and PCB grouping, respectively.



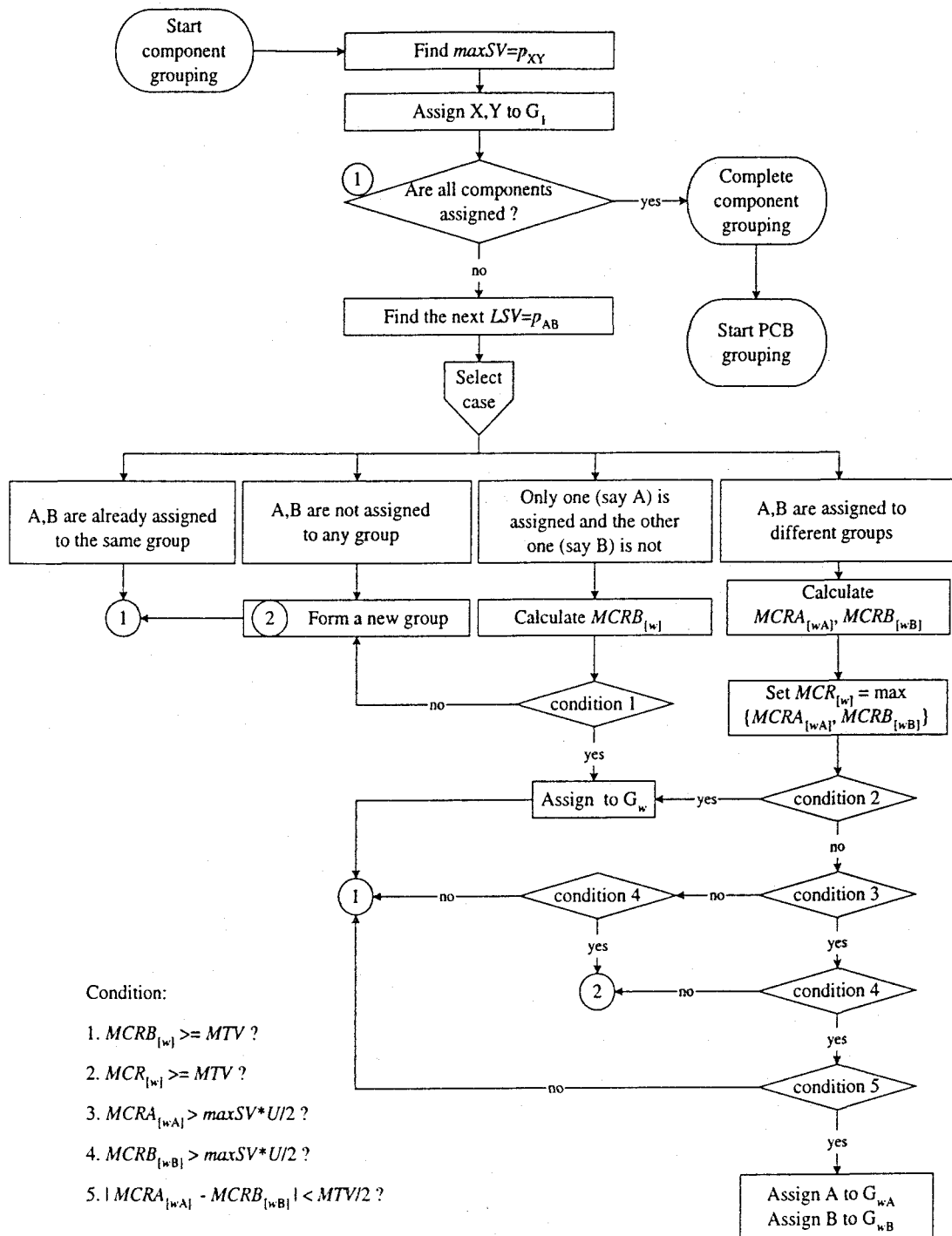


Figure 5.2 Component grouping procedures

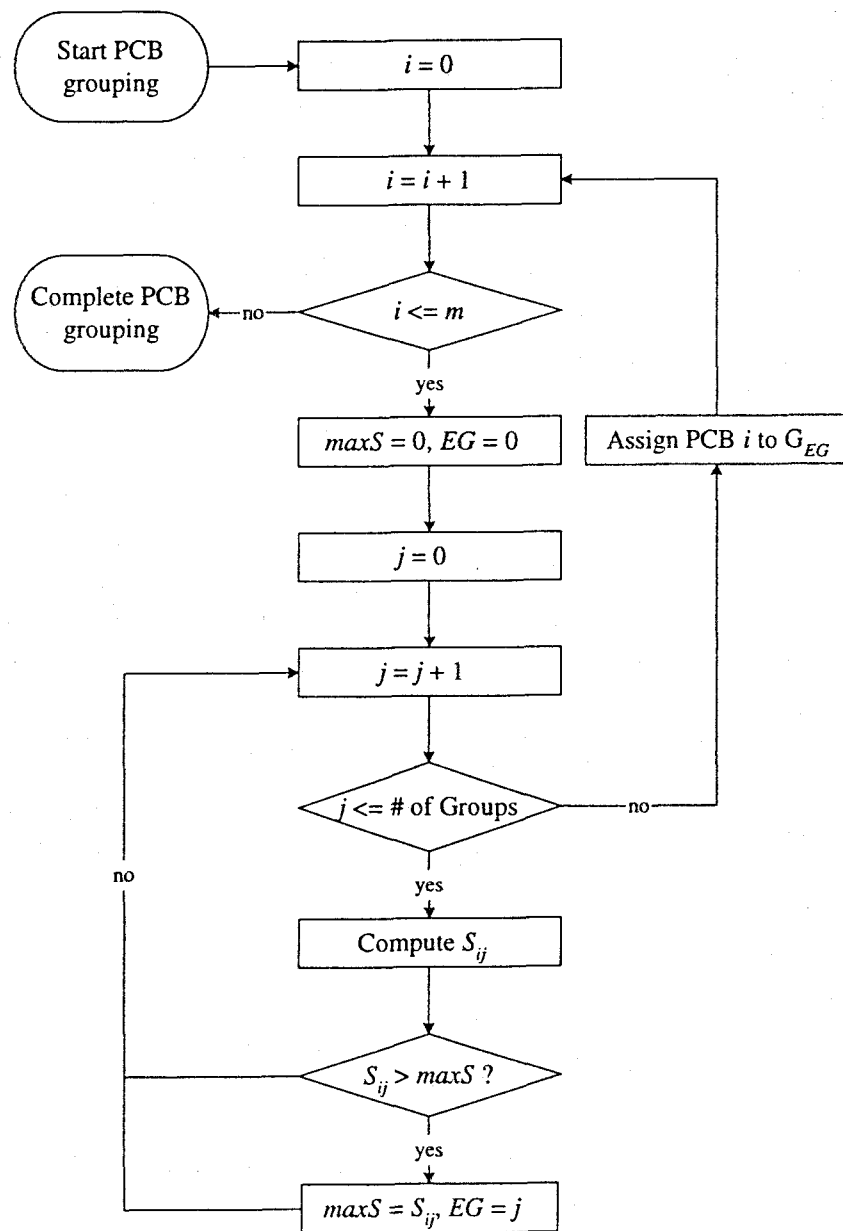


Figure 5.3 PCB grouping procedures

In Figure 5.2, when a pair of components with the largest similarity value is selected from the component-to-component similarity matrix, there are four cases to be considered: both are already assigned to the same group, both are not assigned to any group, only one is assigned to a group, or both are already assigned to different groups. The decision for case 1 is to ignore the current request and check for the next component since both components are already assigned together. Similarly, case 2 is straightforward - a new group is created for the two components.

In case 3, the unassigned component certainly has to be taken care of, but the one that has already been assigned to a group also needs re-evaluation. The threshold  $U$  for allowing a component to join a group is weighted by the similarity value of both components to compare with the maximum closeness ratio of the unassigned component. The closeness ratio evaluates the relationship of the unassigned component with an existing group using the sum of the similarity values to those group members divided by the group size. The group with the maximum closeness ratio becomes the entering group. However, the threshold parameter determines the acceptable minimum closeness ratio value. A high threshold value requires a component to have a very strong relationship with a group in order to join it, otherwise, a new group is preferred. The component that has been assigned is duplicated if a new group is created.

Case 4 is more complicated because both components are already assigned, yet to different groups. Since the current selected pair of components has lower or equal relationship (or similarity value) than the previous pairs, the reassignment is not considered. The possible decisions that can be made are: ignore, create a new group, or assign one or both component(s) to the group(s) where the other one is assigned. Again, the threshold parameter plays a determining role in the decision. Closeness ratios for both components with the groups where the other one exists are calculated. Condition 2 in Figure 5.2 shows that a low threshold value is likely to assign the component with higher maximum closeness ratio to the group where

the other component is assigned. A high threshold value, on the other hand, will either ignore or duplicate both components by new assignments or creating a new group.

In order to use this method effectively and correctly, it is necessary to review the background and objective of this development. This method was first developed for grouping machines in cellular manufacturing [71]. There is a major concern of cost and space for duplicating machines in different groups. The number of groups is a big issue if transportation among cells has a high penalty, thus the use of a threshold value for controlling machine duplication and number of groups. It is understood that in PCB assembly problems, cost and space considerations are not necessary at the grouping stage since the results of interest are the component and PCB groups which are not related to cost or facility space. It is also known that a commonly used component tends to appear in many component groups. Therefore, duplication of components in different groups is permitted. However, the number of groups is still an uncertain factor to the PCB sequence problem. Different threshold levels are used to form different PCB groups to evaluate their performance on the PCB sequencing problem.

### ***5.2.1 Example Illustration***

Throughout this chapter, an example from [72] is used to verify and illustrate the methodology. This example has 45 PCB types using a total 20 component types. Each PCB uses no more than seven component types; thus it is suitable in a case of SMT machine with seven single feeder slots. The PCB and component incidence matrix is listed in Appendix B.2, and the associated component-to-component similarity matrix is in Appendix B.3. The optimal component to double-feeder assignment obtained by using the Hungarian algorithm is shown in Appendix B.4.

The threshold value  $U$  is set to 0.8 and the maximum group size is limited to seven components. The first 10 iterations are listed in Table 5.1 to illustrate the method, followed by the description of some important iterations and calculations in Table 5.2.

Table 5.1 Tabular method for the first 10 iterations

Iteration	LSV	Case	Decision	current group status				
				$G_1$	$G_2$	$G_3$	$G_4$	$G_5$
1	1,5=7	2	create a new group	1,5				
2	2,5=7	3	assign 2 to $G_1$	1,5,2				
3	7,8=7	2	create a new group	"	7,8			
4	6,12=6	2	create a new group	"	"	6,12		
5	8,9=6	3	assign 9 to $G_2$	"	7,8,9	"		
6	8,10=6	3	create a new group	"	"	"	10,8	
7	8,11=6	3	assign 11 to $G_2$	"	7,8,9,11	"	"	
8	8,14=6	3	assign 14 to $G_4$	"	"	"	10,8,14	
9	11,14=6	4	assign 11 to $G_4$ , assign 14 to $G_2$	"	7,8,9,11, 14	"	10,8,14, 11	
10	14,16=6	3	create a new group	"	"	"	"	14,16

LSV = largest similarity value

- Iteration 1. The largest similarity value in the similarity matrix (from top-right to bottom-left; see Appendix B.3) is  $p_{15} = 7$ . Thus  $\max SV$ , as well as current LSV, is set to 7. The first group, denoted  $G_1$ , is created.
- Iteration 2. The next LSV is  $p_{25} = 7$  and case 3 is applied because component 5 is already in  $G_1$  and component 2 has not been assigned. The maximum closeness ratio of component 2 is 6, larger than the minimum threshold value of 5.6. Therefore component 2 is assigned to  $G_1$ .
- Iteration 3.  $p_{78} = 7$  is the next selected pair and case 2 is applied since both components are not assigned to any group. A new group  $G_2$  is created for component 7 and 8.

- Iteration 4.  $p_{6,12} = 6$  is selected and case 2 is applied. A new group  $G_3$  is created because both components are not assigned to any group yet.
- Iteration 5.  $p_{89} = 6$  is the selected pair and case 3 is applied. The entering component 9 has maximal closeness ratio with  $G_2$ . The relationship is significant and larger than the minimum threshold value so that it is assigned to group 2.
- Iteration 6.  $p_{8,10} = 6$ , and case 3 is applied. The entering component 10 has maximal closeness ratio with  $G_2$ , however the relationship is too weak to join group 2. A new group is created.
- Iteration 7.  $p_{8,11} = 6$ , and case 3 is applied. The entering component 11 has maximum closeness ratio 5.33 with  $G_2$  and is greater than  $MTV$ . Component 11 is assigned to  $G_2$ .
- Iteration 8.  $p_{8,14} = 6$ , and case 3 is applied. The entering component 14 has maximum closeness ratio 5.5 with  $G_4$  and is larger than  $MTV$ . Component 14 is assigned to  $G_4$ .
- Iteration 9.  $p_{11,14} = 6$ , and case 4 is applied because both components are already assigned to different groups. First, the closeness ratio for duplicating one component and including it in the others' group is checked. It turns out that the  $MCR$  is not strong enough to duplicate and assign one component to the other. However, their closeness ratios are quite close to the threshold and to each other; thus both components are duplicated and assigned to each other's group.
- Iteration 10. The maximal closeness ratio for entering component 16 is not greater than  $MTV$  so a new group is created.







The PCB grouping procedure is also straightforward. Each board is assigned to the most similar group by comparing the required components and each component group. For example, comparing PCB 1 requirements of components {1,2,3,5} to all component groups, it turns out that group 1 has them all, and therefore PCB 1 is assigned to group 1. The final component and PCB groups are listed in Table 5.3.

Table 5.3 Component and PCB groups

Component group	PCB group
1) 1 5 2 8 9 3 4	1) 1 2 12 31 45
2) 7 8 9 11 14 5 6	2) 3 20 27 40
3) 6 12 4 7 5 8 1	3) 4 16 17 22 38
4) 10 8 14 11 9 6 12	4) 19 32
5) 16 14 11 5 10 12 17	5) 6 15 30
6) 18 9 17 15 16 5 11	6) 8 10 13 21 23 29 35 42
7) 13 11 12 4 5 9	7) 9 11 25 28 33 34 37 39
8) 15 10 14 4 5 6 7	8) 5 7 14 24 26 36 43 44
9) 20 17 19	9) 18 41

### 5.2.2 Discussion

Notice that in the above example, PCB 38 requires identical components as PCB 4, and both boards are assigned to group 3. This is not surprising since identical boards will be grouped in the same manner. Another case is where the component requirements for a board are a subset for another board. For example, the components required by PCB 19, {6,7,8,9,10}, are a subset of requirement of PCB 5, {4,6,7,8,9,10}. Ideally, these two PCBs should be grouped together. However, they are not assigned to the same group in this example. This can happen since the method groups PCBs according to component similarity matrix rather than board similarity matrix. Although this method can not guarantee that PCBs with subset relationships can be assigned to the same group, it is more useful to integrate with

multi-track feeder assignment because both methods are constructed based on the same similarity matrix.

Now, the question is the effect of threshold value on grouping results. Appendix B.5 lists five sets of components and PCB groups obtained by setting different threshold values ranging between 0 to 1. The results show that a higher threshold value tends to create more groups but with fewer components. Notice that PCB 5 and 19 are grouped together in group sets 4 and 5. However, no set with  $U$  less than 0.5 can group them together. This is because components must have very strong relationships to pass high threshold for being grouped together. Whether or not two PCBs with subset relationship can be assigned to the same group depends on the relationships of the non-common components. In this procedure, a high threshold value will discourage the non-common components of the two PCBs to be duplicated in other groups.

Appendix B.5 also shows that the number of groups in set 5 is almost three times larger than sets with threshold value less than 0.5. Two extreme cases that one can imagine are: either one group containing all PCBs or each group containing only one PCB. Both these cases are not desired in PCB sequencing. Therefore the component group size is limited based on machine capacity while connections or similarities between groups are anticipated. The PCB groups in Appendix B.5 will be tested later for determining an appropriate threshold level.

### **5.3 Intra and Inter-Group Sequencing Method (IIGS)**

The PCB sequencing problem becomes straightforward after the grouping stage is accomplished. The boards within a group are likely to use similar components. Sequencing boards within a group by board similarity should result in a minimal setup. Also, choosing the group closest to the current machine setup as the next production group will minimize the inter-group setups.

### 5.3.1 Sequencing Procedure

In this sequencing method, it is suggested that the largest component group with fewer boards be used to initiate the sequence. The associated component group becomes the initial components to be installed on the machine. The board with the most common components to the current setup on the machine is first selected. Ties are broken by choosing the one with least non-common components. The components that have the least usage by the remaining boards within that group are replaced for the next PCB assembly. The subsequent boards follow the same selection criteria until all boards within that group are assembled. The next PCB group is selected in a similar way as if the remaining groups are individual board types. The intra- and inter-group sequencing procedures are repeated until all PCBs are scheduled. Figure 5.4 displays the flowchart for the IIGS sequencing procedure.

### 5.3.2 Example Illustration

The group Set 1 in Appendix B.5 is used in this example and duplicated in Table 5.4. Note that the threshold value,  $U$ , is zero and the maximum component group size is seven. The complete PCB sequence by IIGS and the single feeder setup plan is listed in Appendix B.6.

Table 5.4 Component and PCB groups (duplicated from Appendix B.6, Set 1)

Component Group	PCB Group
1) 1 5 2 8 9 7 11	1) 1 4 9 11 12 14 20 25 28 34 38 45
2) 7 8 9 10 11 14 16	2) 3 6 15 19 27 30 36 40
3) 6 12 4 7 9 18 13	3) 5 16 21 32 33 37 39 43
4) 17 18 3 4 5 8 6	4) 2 17 22 24 31
5) 5 9 10 14 18 16 20	5) 7 8 10 13 18 23 26 29 35 41 42
6) 19 14	6) 44

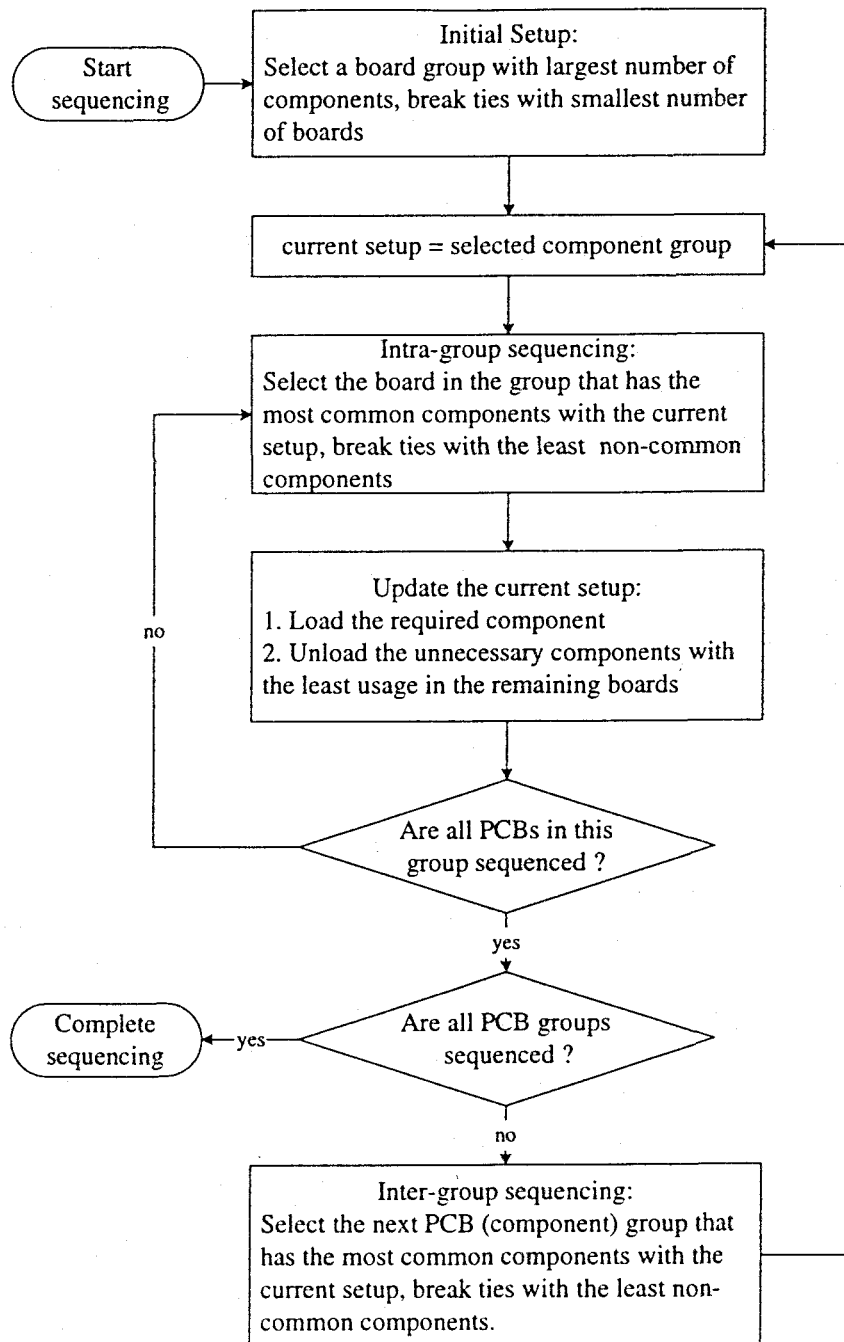


Figure 5.4 IGS PCB sequencing procedures

Component group 4 is selected as the first group due to maximum number of components and least number of PCBs in this group. The components in group 4, {17,18,3,4,5,8,6}, are installed as the initial machine setup. Two boards in group 4 are closest to this setup, PCB 2 and 24. Due to the order of programming, PCB 24 is selected as the first board. Notice that PCB 24 requires component 14 which is not installed in current setup. This requires a machine stop for a feeder setup. The stop downtime depends on the degree of machine automation. Some machines need manual update of feeder information after setup, while advanced machines update database automatically at the time of the setup. If no feeder setup is required, the machine can virtually operate without interruption, or require just minimum adjustments such as changing conveyor width if board sizes are different.

For component 14 is to be installed on the machine, one component must be unloaded from the machine. Sule [72] suggested unloading the components that are not required by the current board and have least usage by the remaining boards within the group. Components 3, 4, and 8 are not used by PCB 24. Among them, component 8 is used by two other boards while components 3 and 4 are needed by three and four boards, respectively. Thus, component 8 is taken off of the line and replaced by component 14.

The next board to be assembled is PCB 2 since it is the most common board to the current setup. Component 17 is unloaded from the machine since it is no longer needed by the remaining boards, and is then replaced by component 2. The remaining boards in group 4 are sequenced in the same manner.

After PCB 17, the last board in group 4, PCB group 1 is selected for next production due to its highest similarity to the current setup. The intra-group sequencing is repeated. The total number of feeder setups is 66, and 41 machine stops are required if no other machine preparation is needed.

### 5.3.3 Discussion

The IIGS method sequences PCBs for minimal setups within and between groups. The procedures are simple, straightforward, and efficient. The component/feeder setup is also planned. However, an optimal switch policy, referred to as *Keep Tool Needed Soonest* policy, will now be used to further improve the sequence plan.

## 5.4 Optimal Component Switch Plan - KTNS

In 1988, Tang and Denardo [75,76] developed a tool change policy called *Keep Tool Needed Soonest* (KTNS) to minimize the number of tool switches for a flexible manufacturing machine. Models and proof of optimality are not duplicated in this research. This policy is described in terms of this research objective as follows:

1. At any instant, components can be installed on to a machine only if they are required by the next PCB.
2. If one or more components have to be installed and no extra space is available, the currently on-line components will be kept in the order of soonest need.

### 5.4.1 KTNS Procedure

The KTNS policy is not used to find the optimal PCB sequence but to plan for the optimal component switches for a specific sequence. The flowchart of KTNS procedure and pseudo-code are displayed in Figure 5.5 and Appendix B.7, and described as follows:

Step 0. **Initialization:** Denote  $i$  as the sequence index,  $i = 1$ , and  $E_1(k)$  the current status of component  $k$ , 1 is on-line, 0 is off-line, and -1 is on-line but not used by the remaining sequenced boards. Similarly,  $E_2(k)$  represents the component status for the next sequenced PCB. Before starting the assembly, assume that all components are installed on the machine initially without concerning with the slot space. Therefore  $E_1(\bullet)$  for all  $k$  are initially set to 1.

Step 1. **Keep the components required by the next PCB:** If there is no PCB left then the procedure is completed. The components that are required by the first sequenced board ( $B_1$ ) are kept at this stage, thus  $E_2(\bullet)$  is set as the same board incidence vector  $a_{B_1}$  from the incidence matrix. The number of components to be kept,  $KN$ , besides the next board required components,  $RB_i$ , is the number of available slot spaces,  $sn$ , minus  $RB_i$ . If  $B_i$  is the last PCB, the situation is different and there is a plan for last PCB (go to Step 5). If  $B_i$  is not the last PCB and the next PCB needs all slot spaces ( $KN = 0$ ), then go to Step 6 for feeder setup plan. If the next PCB does not use all slot spaces ( $KN > 0$ ), then go to Step 2 and search the components needed soonest.

Step 2. **Keep the components needed soonest:** If  $B_i$  is not the last PCB, then starting from  $B_{i+1}$  those components are kept which are currently on-line but not on the list of next setup based on the urgency of need. Every time a component is kept,  $KN$  is decreased by one. The search is repeated until either  $KN = 0$  (go to Step 6) or the last PCB is encountered (go to Step 3).

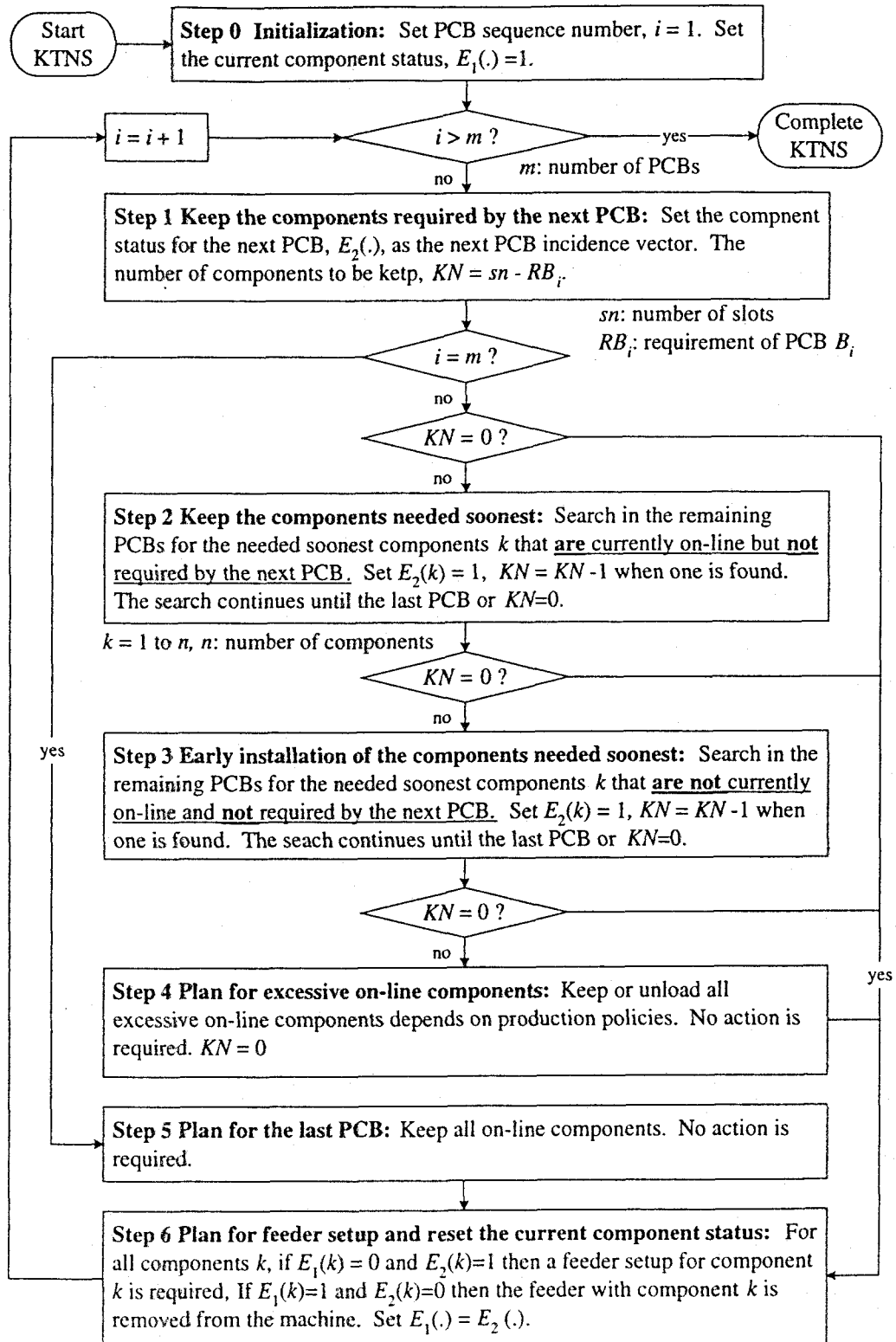


Figure 5.5 KTNS procedures for single feeder configuration



**Step 3. Early installation of the components needed soonest:** Occasionally, especially near the end of production, some components that are currently on-line are no longer used by the remaining boards; thus there is no reason to keep them. On the other hand, some other components that are not currently on-line may be required by subsequent boards but not the next board. If  $KN$  is not zero after Step 2, it means that there are empty slots where the components can be installed for future use. This early installation of components will not increase the number of setups because the components will be needed eventually. The selection of components for early installation is also based on the rule of *needed soonest*. The search is again repeated until either  $KN = 0$  (go to Step 6) or the last PCB is encountered (go to Step 4).

**Step 4. Plan for excessive components:** A more extreme case for  $KN$  being not zero is that the current setup can be used to process the remaining boards. Depending on production policies, these components can be either left on the machine or unloaded from the machine. In this research, those components that are not required by the remaining boards are marked as  $-1$  and not counted as setups. This step ensures that  $KN$  is zero. Go to Step 6.

**Step 5. Plan for the last PCB:** No extra action is needed to setup for the last PCB as long as the required components are on-line. For computation purposes, these on-line components are kept whether they are used or not.

**Step 6. Plan for feeder setup and reset the current component status:** The component status  $E_1(\bullet)$  and  $E_2(\bullet)$  is compared. A feeder setup is required if  $E_1(k) = 0$  and  $E_2(k) = 1$ , and component  $k$  is removed

from the machine if  $E_1(k) = 1$  and  $E_2(k) = 0$ .  $E_2(\bullet)$  is assigned to  $E_1(\bullet)$ . Set  $i = i + 1$  for next sequenced PCB and repeat Step 1.

#### 5.4.2 Modified KTNS for Multi-track feeder Configuration

The KTNS policy is modified to plan for multi-track feeder systems. The flowchart and pseudo-code of modified KTNS for double feeder case is displayed in Figure 5.6 and Appendix B.8, respectively.

The procedures for modified KTNS are similar to the original policy except the components that are assigned to the same feeder are loaded or unloaded at the same time. Notice that  $sn^* = sn \times r$ , and the requirement of  $RB_i^*$  is calculated as

$$RB_i^* = \sum_{k=1}^n (a_{B_i,k} \text{ or } a_{B_iC(k)})$$

The rules of keeping components follow the order below:

1. The components required by the next PCB and the components that are assigned to the same feeders.
2. The components that are needed soonest and the components that are assigned to the same feeders.
3. If there are empty slots, the components that are needed soonest and the components that are assigned to the same feeders can be installed prior to production.
4. If there are empty slots and the current setup can be used for the remaining assemblies, the components can be either kept on machine or removed from machine depending on the production policies.

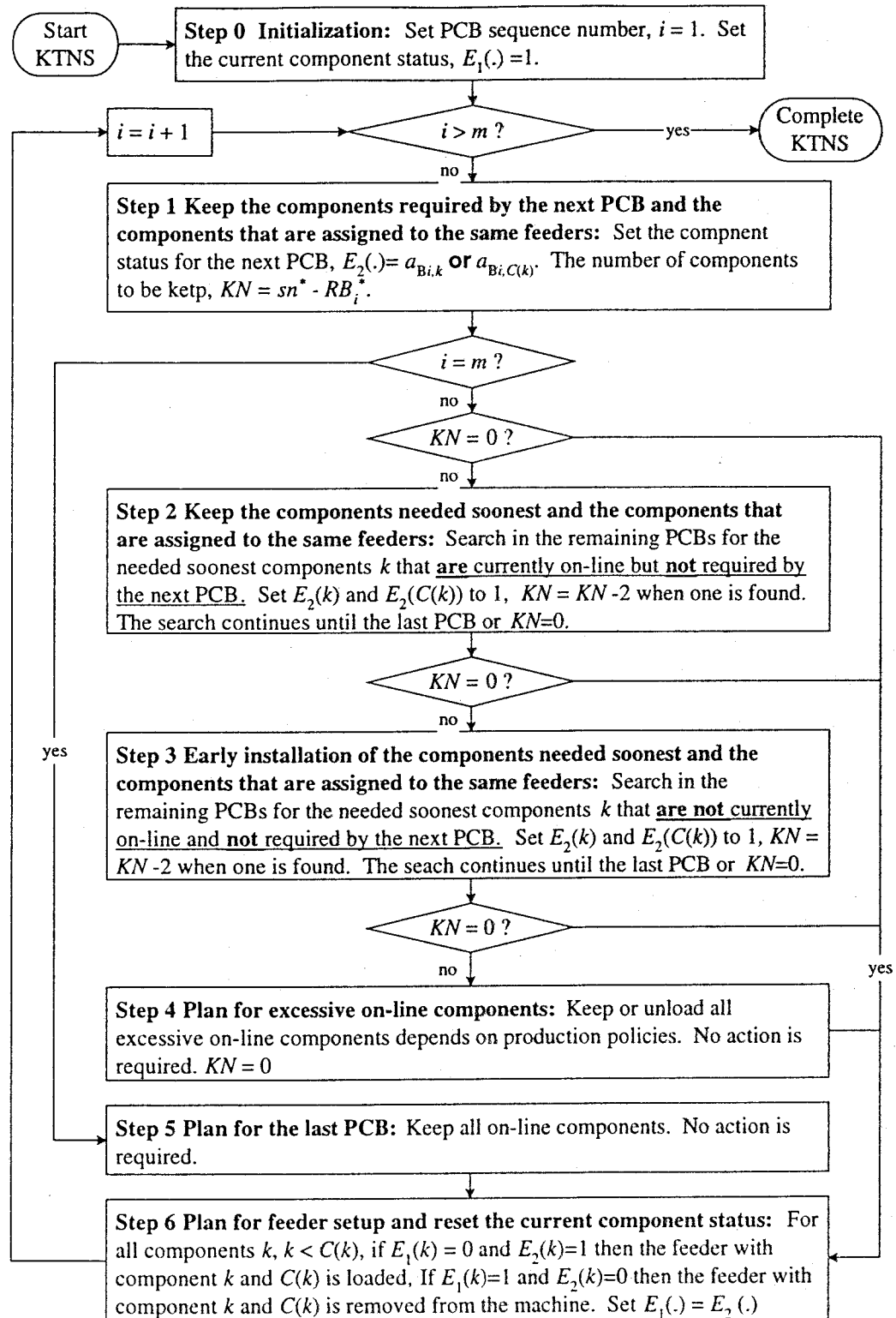


Figure 5.6 Modified KTNS procedures for double-feeder systems.

## 5.5 Application illustration

The PCB sequence obtained from previous section is used to demonstrate the implementation of KTNS and the modified KTNS policies. The complete component switch plans for single and double feeder cases are listed in Appendix B.9 and B.10.

### 5.5.1 Single Feeder Setup Plan

#### Initialization:

Let  $E_1(\bullet)$  be initially set to 1 for all  $k$ . The components that are required by PCB 24 are first kept.  $E_2(\bullet)$  is set as the same value as the vector  $a_{24}$  from the incidence matrix.

#### Keep the components needed soonest:

Iteration 1.  $KN = sn - RB_I = 7 - 5 = 2$ , only two components can be kept from  $E_1$ . As shown in Table 5.5, component 2 and 3 used by PCB 2 are needed soonest components because  $E_1(2) = E_1(3) = 1$  and not used in PCB 24. Therefore the starting setup is  $\{2,3,5,6,14,17,18\}$ .

Iteration 2. In this iteration, two components are to be kept and the needed soonest components are 14 and 18 used by PCB 17 and 35, respectively.

Iteration 3. PCB 22 requires components 1,3,4,5, and 7. This leaves two components to be kept. Component 14 and 2 used by PCB 17 and 34 are kept for being needed soonest.

Iteration 45. For the last iteration, the on-line components are kept.

Table 5.5 Illustration of KTNS procedure

		(iteration 1) components																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$E_2$		1	1			1	1								1			1	1		
24		0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0
2		0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22		1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

n

		(iteration 2) components																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$		0	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0
$E_2$			1	1	1	1	1								1				1		
2		0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22		1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
31		1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
17		0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
34		0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0
25		0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0
28		1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0
28		1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0
4		1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
20		0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
45		1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0

n

		(iteration 3) components																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$		0	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
$E_2$		1	1	1	1	1		1							1						
22		1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
31		1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
17		0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
34		0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0

n

		(iteration 45) components																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$		1	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0
$E_2$		1		1	1						1			1	1	1					
44		0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0

### 5.5.2 Double-Feeder Setup Plan

The optimal component to double-feeder assignment is: {(1,5) (2,9) (3,4) (6,12) (7,8) (10,15) (11,13) (14,16) (17,18) (19,20)}

#### Initialization:

$E_1(\bullet)$  are set to 1 just like in the previous example. The components required by PCB 24 are first kept. The value in  $E_2(\bullet)$  for  $a_{24}$ , and the associated components assigned to the same feeders are set to 1.

#### Keep the components needed soonest:

The number of components that can be kept is calculated as:

$$sn^* = sn \times 2 = 14, \text{ and,}$$

$$RB_1^* = \sum_{k=1}^{20} (a_{24k} \text{ or } a_{24C(k)}) = 8$$

$$\text{Thus, } KN = sn^* - RB_1^* = 14 - 8 = 6$$

Iteration 1. Six components can be kept from  $E_1$ . As shown in Table 5.6, component 2, 3 and 4 used by PCB 2 are selected because they are *needed soonest*. Component 9 is kept because it is assigned with component 2. Component 7 and component 8 used by PCB 22 and PCB 31 respectively are also *needed soonest*. Notice that they are also assigned to the same feeder.

Iteration 2. Components 2, 3, 4, 5, and 6 enter the setup list for being used by PCB 2. Their associated assigned components 1, 9, and 12 also enter the setup list. Six more components can be kept from  $E_1$ . Components 7 and 8, which are both needed soonest and assigned to the same feeder, enter the setup first, followed by components 14 and 16, which are also in the same feeder. Component 18 is kept for needed soonest along with 17, its assigned component.

Table 5.6 Illustration of modified KTNS procedure on double feeder case

(iteration 0) components																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$E_2$	$I^*$				1	1						$I^*$			1		$I^*$		1	1
24	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0

\*: the components that are assigned to the same double feeder

(iteration 1) components																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$E_2$	$I$	1	1	1	1	1	1	1	$I^*$			$I$		1		$I$	1	1		
24	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0
2	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
31	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

||

(iteration 2) components																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$E_1$	1	1	1	1	1	1	1	1	1	0	0	1	0	1	0	1	1	1	0	0
$E_2$	1	1	1	1	1	1	1	1	$I$			$I$		1		1	$I$	1		
2	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
31	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
34	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0
25	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0
28	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0
28	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0
4	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
20	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
45	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0

||

## 5.6 Evaluation of the Methodology

The methods developed are evaluated using the research data and the industry data presented in chapter 4. The first set of evaluation experiments uses the previous example to illustrate the effect of the threshold value on the results. The threshold value is known to form different PCB groups and is suspected to affect the final results. The five sets formed by different threshold values in Appendix B.5 are evaluated by the integration method developed so far. The evaluation is based on a seven-slot machine with double feeder option. The second evaluation is performed on the industry data set. Both evaluations are tested on a Dell 300MHz PC with 64 MB RAM with Microsoft NT 4.0 operation system. The method combinations used for effect estimations are:

- M1. IIGS sequence without KTNS policy (IIGS),
- M2. Random Board Sequence with KTNS policy (RBS/KTNS),
- M3. IIGS sequence with KTNS policy ( IIGS/KTNS),
- M4. IIGS sequence with modified KTNS and Optimal Feeder Assignment (IIGS/OFA DBLKTNS),
- M5. Random board sequence with modified KTNS and optimal assignment (RBS/OFA DBLKTNS),
- M6. IIGS sequence with modified KTNS but Random Feeder Assignment (IIGS/RFA DBLKTNS), and
- M7. Random sequence and assignment with modified KTNS (RBS/RFA DBLKTNS)

The main and interaction effects and interactions of the evaluated method combinations are summarized in Table 5.7. Ten random PCB sequences and double feeder assignments are generated to be used in random cases.



Table 5.7 Main and interaction effects of evaluated methods

Method	IIGS	KTNS	MKTNS	OFA	Main effect	2-way interactions	3-way interactions
M1	•				IIGS		
M2		•			KTNS		
M3	•	•			IIGS, KTNS	IIGS*KTNS	
M4	•		•	•	IIGS, MKTNS, OFA	IIGS*MKTNS, IIGS*OFA, MKTNS*OFA	IIGS*MKTNS*OFA
M5			•	•	MKTNS, OFA	MKTNS*OFA	
M6	•		•		IIGS, MKTNS	IIGS*MKTNS	
M7			•		MKTNS		

MKTNS: Modified KTNS

It is impossible to estimate the main effects for KTNS and IIGS methods because there is no baseline or worst case to be compared with. Furthermore, the integration effect of both methods involves two-way interactions. Therefore, comparing the main effects alone is meaningless. Table 5.8 lists the comparisons that can be interpreted from Table 5.7.

Table 5.8 Interpretation of comparisons from evaluation

Effect	Comparison	Interpretation
E1	M3 – M1	Improvement made by KTNS based on IIGS
E2	M3 – M2	Improvement made by IIGS based on KTNS
E3	M7 – M6	Improvement made by IIGS based on MKTNS
E4	M7 – M5	Improvement made by OFA based on MKTNS
E5	M2 – M7	Effects of Double vs. Single feeder based on KTNS
E6	M2 – M5	Effects of Double vs. Single feeder based on OFA/KTNS
E7	M3 – M4	Effects of Double vs. Single feeder based on OFA/KTNS/IIGS
E8	M7 – M4	Effects of integration of OFA and IIGS based on MKTNS
E9	M2 – M4	Best vs. Worst production capability

### 5.6.1 Evaluation on Example Data

Since M2, M5, and M7 use random PCB sequences, they will not be affected by threshold values and their results are separated from the other four methods. The complete results are given in Appendix B.11.

With this example, there is no conclusive evidence that the threshold level has a correlation with the final results, except that the computation time using IIGS method decreases as the threshold value increases. Figure 5.7 shows the results of feeder setups and required machine stops in relation to method and threshold values. It can be observed that the effect of variation for each method is insignificant for different threshold levels and that the method effects are similar for all threshold values. Thus, the confounding effect of threshold level can be eliminated. The numerical results are summarized in Table 5.9.

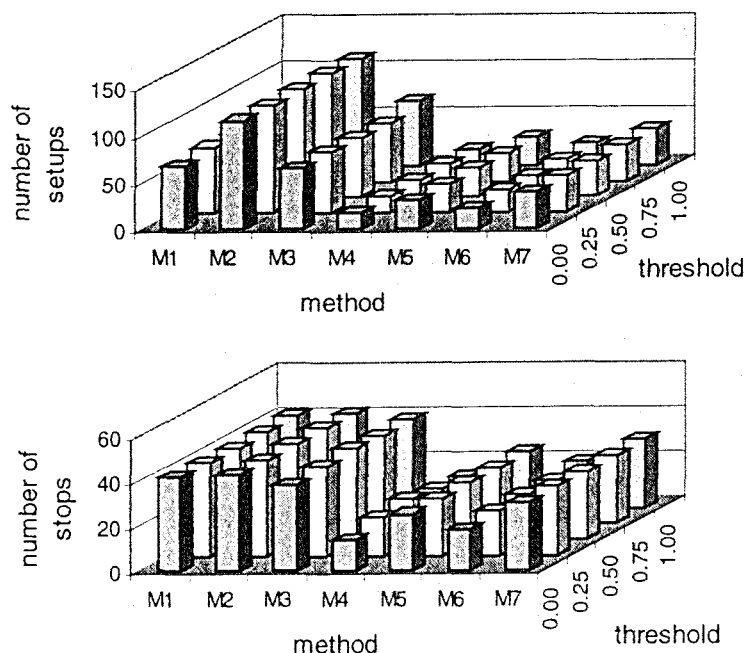


Figure 5.7 Evaluation results

Table 5.9 Results from example data

Summarized results			
method combination	feeder setups	machine stops	
M1	69.4	40.8	
M2	114.6	42.1	
M3	64.4	39.0	
M4	17.4	15.8	
M5	30.1	24.8	
M6	22.5	19.4	
M7	38.2	29.9	

Effect	Comparison	% reduction in	
		setups	stops
E1	M1-M3	7.2	4.4
E2	M2-M3	43.8	7.4
E3	M7-M6	41.0	35.3
E4	M7-M5	21.2	17.1
E5	M2-M7	66.7	29.0
E6	M2-M5	73.7	41.1
E7	M3-M4	73.0	59.5
E8	M7-M4	54.5	47.2
E9	M2-M4	84.8	62.5

By comparing M1 and M3 (E1), the effect of the KTNS is estimated to improve the component switch plan developed by Sule, approximately 7% for feeder setup and 4% for reduction in machine stops. On the other hand, the effect of IIGS method for single feeder system, based on KTNS policy and estimated by comparing M2 and M3 (E2), results in reductions of 44% and 7% in setup and machine stops, respectively. This indicates that the improvement made by IIGS method is more substantial than the KTNS policy.

The use of IIGS with modified KTNS policy (E3) results in a reduction of about 41% in feeder setups and 35% in required machine stops. This result shows that IIGS significantly reduces machine stops in the double-feeder system; however, the setup reduction rate is the same as in the single feeder system.

Comparison of M5 and M7 (E4) shows that the optimal feeder assignment with modified KTNS policy explains 21.2% of feeder setups and 17.1% of machine stops reduction.

The effect of double-feeder with KTNS (E5) significantly reduces setups (67%) and stops (29%). By applying optimal assignment (E6), an additional reduction of 7% in setups and 12% in stops is achieved. Although the magnitude of setup reduction is not affected by applying IIGS method (E7), the overall stops are dramatically decreased, an additional reduction of 18%.

Impressively, the integration of optimal component-to-feeder assignment and IIGS sequencing method (E8) results in over half of feeder setup reductions and machine stops. Effect E9 shows that a company utilizing this plan can be expected to outperform its opponents who do not have the multi-track feeder capability with 85% reduction on feeder setups and 63% on machine stops.

### **5.6.2 Evaluation Using Industry Data**

The seven data sets from industry in chapter 4 are used to evaluate the performance of the methodology described here. The number of slots is 150. Ten sets of random board sequences and random component-to-feeder assignments are generated for each random case. Since threshold value has no significant effect on feeder setups and machine stops, the threshold is set to 0.5 for component and PCB grouping. Table 5.10 summarizes the results; details are given in Appendix B.12.

The results show that the use of sequencing methods E2 and E3 is more effective with industry data. Comparing double-feeder and single-feeder setups (E9), the results are consistent for both data sets with about 85% feeder setup and 62% machine stop reduction. The major differences with the industry data sets include:

Table 5.10 Results from example data (section 5.6.1) and industry data sets

Effect	% reduction in feeder setup		% reduction in machine stop	
	example	industry	example	industry
E1 Improvement by KTNS based on IIGS	7.2	47.1	4.4	22.0
E2 Improvement by IIGS based on KTNS	43.8	60.1	7.4	46.7
E3 Improvement by IIGS based on MKTNS	41.0	58.7	35.3	48.5
E4 Improvement by OFA w/ MKTNS	21.2	59.4	17.1	23.4
E5 Double w/ MKTNS vs. Single w/ KTNS	66.7	37.0	29.0	8.1
E6 Double w/ OFA/MKTNS vs. Single w/ KTNS	73.7	74.4	41.1	29.6
E7 Double w/OFA/MKTNS/IIGS vs. Single w/KTNS	73.0	66.0	59.5	28.9
E8 Integration of OFA and IIGS w/ MKTNS	54.5	78.5	47.2	58.8
E9 Best vs. Worst production capability	84.8	86.5	62.5	62.1

1. The KTNS policy works more effectively with industrial cases (E1). The effects are augmented because a large amount of components and PCB types are involved.
2. The IIGS method significantly reduces the machine stops in the single-feeder case (E2). As expected, IIGS groups PCB by component similarities, and sequences PCBs according to the board similarity within and between groups. The results are also augmented by the problem size and component and PCB characteristics.
3. Although the use of double feeders with KTNS policy contributes 37% to feeder setup reduction (E5), the effect is only half of the outcome with the example case. Notice that the double feeder assignment (E4) helps to recover the loss. The effect of the double-feeder integrated with assignment method are close for both cases, within 10%.

## 5.7 Chapter Summary and Conclusions

This chapter presented a methodology that integrates the grouping procedures, intra- and inter-group sequencing procedures, and KTNS policy to approach a PCB sequence and a feeder setup plan so that the feeder setups can be significantly reduced. It also describes the changes made to this procedure for solving the multi-track feeder assignment and the sequencing problem.

A published data set was used to verify the methodology and investigate different solution alternatives. Several industry data sets were also used to test the performance of the methodology. The following conclusions are drawn from this chapter:

1. The evaluation results show that the integrated methodology reduces not only the feeder setup but also the required machine stops significantly.
2. The threshold value that determines the component and PCB groups has no significant effect on final results in terms of feeder setups or machine stops. The final results are dependent on the methods applied and also on the interaction of each component.
3. The effect of each method is dependent on the problem size, representing the number of components and PCB types, and interaction with other factors.
4. The efficiency of the integrated methodology also depends on the problem size. The industry data sets are all solved within acceptable CPU times ranging from 116 seconds to 888 seconds for PCB types ranging from 281 to 843 and component types ranging from 548 to 826.

## CHAPTER 6 EXPERIMENTATION AND ANALYSIS

As shown in the previous chapters, the performance of heuristic algorithms is affected by the data distribution and the interaction of the component to feeder assignment and board sequencing problems. This chapter details an experiment designed to investigate the effects of these confounding factors on machine setup.

The experiment starts with an analysis of industry data. After understanding data characteristics, a random number generator is developed to generate experimental data sets. A simulation experiment is conducted to evaluate the data and statistical analyses are performed to identify significant effects.

### 6.1 Industry Data Analysis

In order to fully understand the performance of the methodology in different production environments, the industry data set is thoroughly analyzed. First, the PCB and component incidence matrix is separated into two sections: the PCB requirement, and component usage frequency. The PCB requirement (REQ) is the total number of component types required for a PCB assembly and equals the sum in the incidence matrix rows. The component usage frequency (USG) is the total number of PCBs that need a particular component type and is the sum of the matrix columns. The description of the industry data set is given in Table 6.1. Notice that each component type is used by at least one PCB, and each PCB needs at least one component type. Table 6.1 shows that, on average, each component type is needed by 25 PCBs, and each PCB requires about 32 components. Another measure is the matrix density, which is the sum of the incidence matrix (or total number of 1's) divided by the product of number of rows and columns. The USG and REQ histograms for industry data C are shown in Figure 6.1. It can be seen that the distributions of both histograms are skewed to the left.

Table 6.1 Industry data set descriptions

	Data Set							average
	A	B	C	D	E	F	G	
PCBs ( <i>m</i> )	437	458	281	744	620	608	843	570
Components ( <i>n</i> )	695	625	547	796	769	712	826	710
density % ( <i>d</i> )	4.6	4.8	6.2	3.9	4.3	4.4	3.8	4.6
minimum USG	1	1	1	1	1	1	1	1
mean USG	20.2	21.9	17.5	28.7	26.8	26.5	31.7	24.8
maximum USG	218	229	148	350	315	289	396	277.9
std. dev. USG	35.2	37.8	27.3	54.0	49.7	47.6	60.8	44.6
minimum REQ	1	1	1	1	1	1	1	1
mean REQ	32.1	29.9	34.1	30.7	33.3	31.0	31.0	31.7
maximum REQ	144	106	106	145	144	111	146	128.9
std. dev. REQ	24.2	25.2	25.1	24.9	24.8	25.3	25.1	24.9

density: (total number of 1's in incidence matrix) / ( $m \times n$ )  
USG: component usage frequency  
REQ: PCB requirement

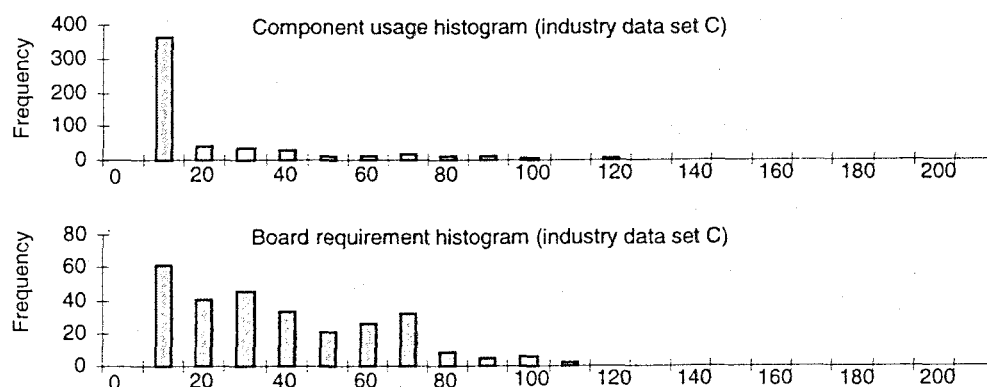


Figure 6.1 The component usage and board requirement histograms for Data C

Because of the direct connection to the component-to-component similarity matrix used in multi-track feeder assignment and component grouping, variations in USG and REQ may contribute to the differences between the example data and industry data in Chapter 5. Further analysis requires more data with different



distributions. To provide for additional variability and variety, simulated data are created by a random number generator for use in the experiment.

## **6.2 Description of the Simulation Data**

The simulation data used in this experimentation are generated by a random number generator. With control over the distribution parameters, the data sets cover extreme cases so that the methodology can be evaluated. The four extreme cases evaluated are:

- EHH: High USG and REQ variations
- EHL: High USG variation and Low REQ variation
- ELH: Low USG variation and High REQ variation
- ELL: Low USG and REQ variations

### **6.2.1 *Random Number Generator for Incidence Matrix***

The random number generator is constructed so that both the USG and REQ frequency distribution parameters can be controlled. The inputs include:

1. problem size,
2. matrix density,
3. minimum USG and REQ,
4. mean USG and REQ,
5. maximum USG and REQ,
6. standard deviation of USG and REQ, and
7. random number seed.

An exponential random number generator is used to generate random numbers for both USG and REQ frequency distributions because of the similar distribution shapes (Figure 6.1). The equation used is:

$$\Phi^* = -\sigma \times \log(\Phi) + \mu \quad (6.1)$$

where  $\Phi \in (0, 1]$  is uniformly distributed random number generated by the system, and  $\mu$  and  $\sigma$  are input mean and standard deviation.  $\Phi^*$  is the generated random number; this is further filtered for minimum and maximum restrictions. One row and one column vector of random numbers are generated for USG and REQ frequency distributions separately. These two frequency vectors are adjusted to meet input density value and then used to construct the incidence matrix. Notice that in equation (6.1) if the input  $\sigma$  is zero, all generated random numbers will be constant and equal to  $\mu$ . This means an equal probability of component types to be used by PCBs and vice versa.

### 6.2.2 Experiment Data Configurations

The experiment data are generated based on the same data size so that the size effect can be ruled out. Data density is set to the typical industry level and fixed. Industry data C (see Table 6.1) is chosen as the control set to compare with the simulation data sets. The first data set, EMM, simulates the industry data C to verify that the random number generator works as expected. Then two sets of four extreme cases are generated with different random seeds. The configurations of the experimentation data sets are listed in Table 6.2. The maximum of REQ frequency is controlled under 150, which is the number of slots. Both USG and REQ means are set to the same value as in data C.

Table 6.2 Experimentation data set descriptions

Exp#	<i>m</i>	<i>n</i>	density	USG				REQ			
				min	max	mean	stdev.	min	max	mean	stdev.
Data C	281	548	0.0622	1	148	17.48	27.1	1	106	34.09	24.1
EMM1	281	548	0.0622	1	147	17.48	27.5	1	107	34.09	22.5
EHH1	281	548	0.0622	1	131	17.48	30.7	1	141	34.09	43.5
EHL1	281	548	0.0622	1	115	17.48	30.3	34	36	34.09	0.29
ELH1	281	548	0.0622	17	20	17.48	0.66	1	136	34.09	41.9
ELL1	281	548	0.0622	17	21	17.48	0.66	34	36	34.09	0.29
EMM2	281	548	0.0622	1	167	17.48	27.4	2	107	34.09	22.1
EHH2	281	548	0.0622	1	118	17.48	30.1	1	139	34.09	41.2
EHL2	281	548	0.0622	1	188	17.48	32.4	34	35	34.09	0.28
ELH2	281	548	0.0622	17	21	17.48	0.69	1	144	34.09	41.5
ELL2	281	548	0.0622	17	21	17.48	0.68	34	36	34.09	0.28

The key parameters are the USG and REQ standard deviations. The high level is set to a large number and the low level to zero. The actual standard deviations are determined by the random numbers and the input means. The actual USG and REQ minimum and maximum values are dependent on the actual standard deviations and the input minimum and maximum values. The USG and REQ histograms for EMM are shown in Figure 6.2 whose distributions are close to the industry data. Figure 6.3 displays an illustration of histograms for the extreme case EHL. The data scatter plots are shown in Figure 6.4 for visual comparison.

In Figure 6.4, each dark spot represents the use of a component by a PCB. Figure 6.4a shows vertical patterns in data C, which means that some components are commonly used by many PCBs. It also shows horizontal patterns meaning that some PCBs use lots of components. Figure 6.4c shows strong patterns both ways, which means that the design and requirements of these PCBs are similar. Because the data density is fixed, if there exists some frequently used components, then there must be some rarely used ones to balance the average. Figure 6.4d and 6.4e verify that the high variation of USG and REQ contributes to the vertical and horizontal patterns, respectively. Figure 6.4f shows no pattern at all with a random distribution of components among PCBs.

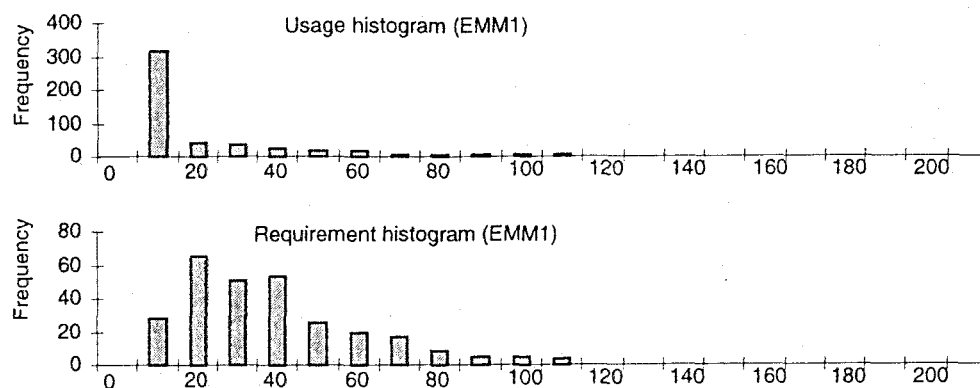


Figure 6.2 The USG and REQ histograms for industry simulated data EMM.

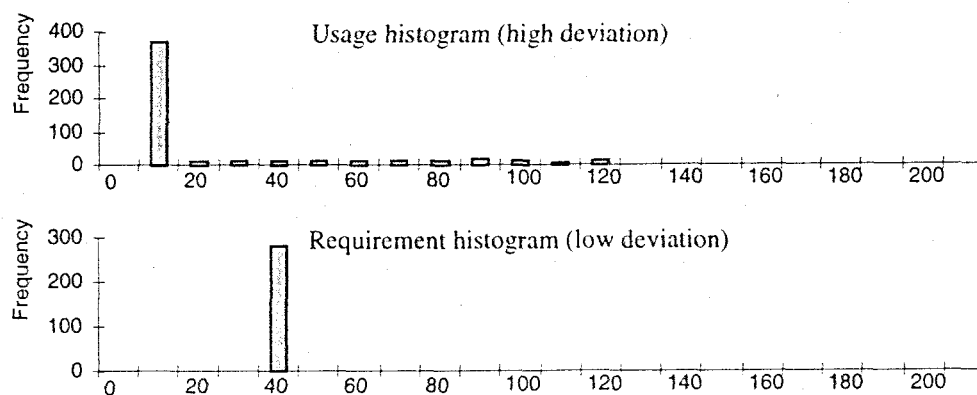


Figure 6.3 Example of high deviation USG and low deviation REQ (EHL).

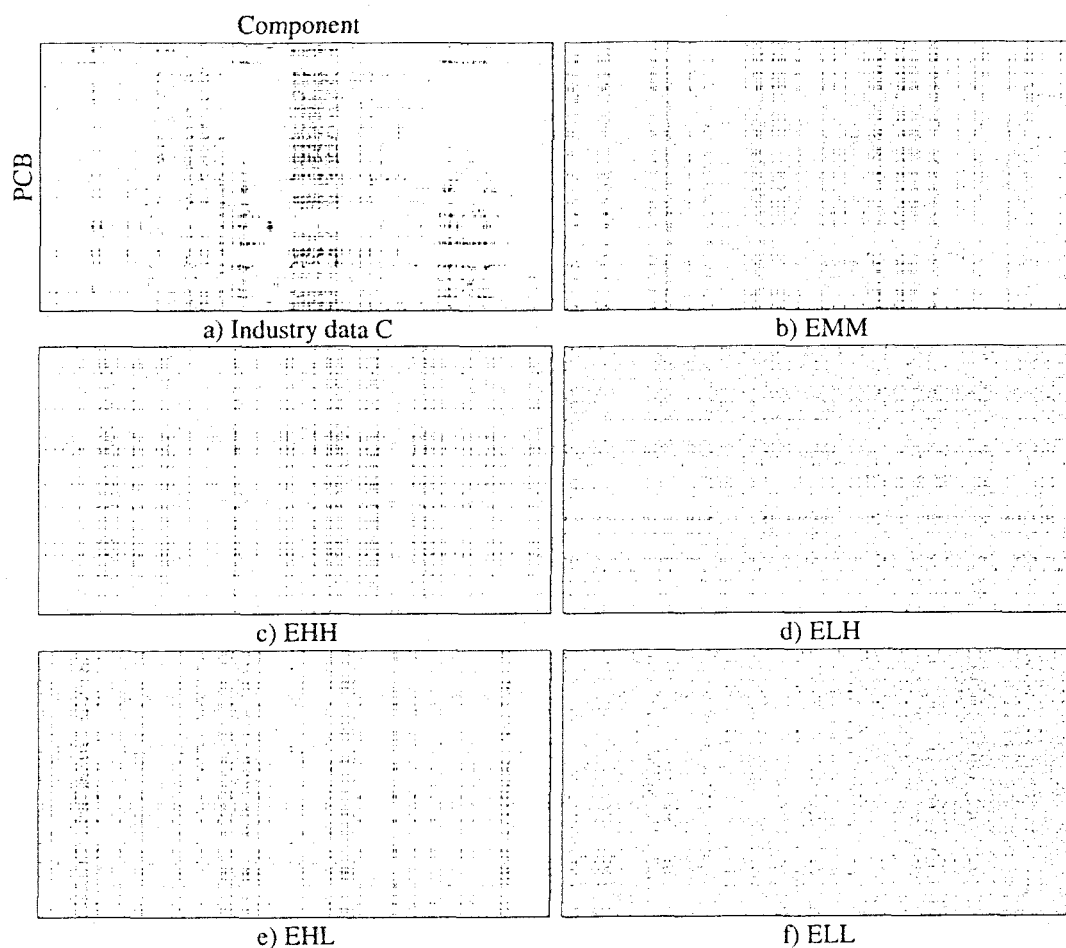


Figure 6.4 Visual comparisons of industry data and simulation data

### 6.3 Experiment Design and Results

The four factors that are considered in this experimentation are:

- (A) Component usage variation,
- (B) PCB requirement variation,
- (C) PCB sequence, and
- (D) Component to multi-track feeder assignment.

A  $2^4$  full factorial experiment is designed to investigate the main effects of these four factors and their interactions on the number of feeder setup and the solution time (CPU times) for feeder setup planning. Since the number of machine stops is dependent on feeder setup, it is excluded from the analysis. The two simulated data sets provide replication with more degrees of freedom. The experiment design is summarized in Table 6.3. The results are given in Appendix C.1.

Table 6.3 Summary of  $2^4$  factorial design

Number of experimental factors	4	Factors	Low Value	High Value
Number of blocks	2	(A) USG stdev.	low (-1)	high (+1)
Number of responses	2	(B) REQ stdev.	low (-1)	high (+1)
Number of runs	32	(C) Sequence	RBS (-1)	IIGS (+1)
Error degrees of freedom	20	(D) Assignment	RFA (-1)	OFA (+1)
Randomized	No	RBS: random board sequence		
		IIGS: intra- and inter-group sequence		
		RFA: random feeder assignment		
		OFA: optimal or best feeder assignment		

### 6.3.1 Experiment Results

Based on the plots of residuals versus predicted values, data for CPU time satisfies the assumptions of normality and equality of variance. The residual plot of feeder setup shows a normality problem, thus requiring data transformation. A log transformation is used; Figure 6.5 shows the residual plots of feeder setup before and after the data transformation.

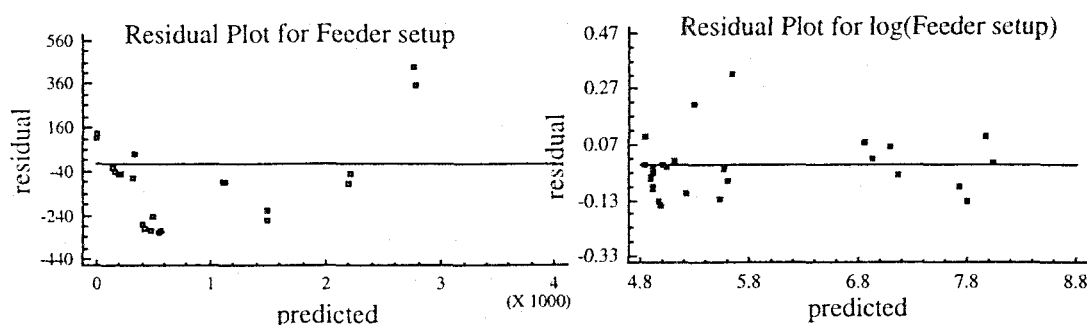


Figure 6.5 Data transformation for Feeder setup

### 6.3.2 Analysis of Effects on Feeder Setup

The analysis starts with full model since the interaction effects are expected. The analysis of variance on four-way interaction shows that the interaction ABCD is insignificant for both responses. The three way interaction ANOVA for transformed data  $\log(\text{Feeder setup})$  is shown in Appendix C.2.

The ANOVA shows the effect ABD is significant at 99% confidence level along with AB, AC, and CD interactions. The main effect B, the PCB requirement variation, shows no significant effect on the response. The adjusted R-squared statistic indicates that the 3-way full model explains 99.2% of data variability in  $\log(\text{Feeder setup})$ . In the reduced model, all the insignificant effects are dropped. The three way interaction ABD turns out to be insignificant ( $P\text{-value} = 0.023$ ). Another reduced model which drops the ABD effect shows the effect AB to be borderline ( $P\text{-value} = 0.01$ ). The backward selection stepwise regression method is then used to remove insignificant effects at 99% confidence level. The selection steps are shown in Appendix C.3. The final model, displayed in Table 6.4, shows the PCB requirement variation removed from the model along with its interactions. The interaction terms that are significant among the three main effects are AC and

CD. The adjusted R-squared value indicates that the fitted model explains 98.06% of the data variability.

Table 6.4 Final fitted model for log(Feeder setup).

Parameter	Estimate	Standard Error	T Statistic	P-Value
CONSTANT	5.63435	0.0276124	204.051	0.0000
A:USG stdev.	-0.59261	0.0276124	-21.4616	0.0000
C:Sequence	-0.70812	0.0276124	-25.6448	0.0000
D:Assignment	-0.24797	0.0276124	-8.98041	0.0000
AC	0.51488	0.0276124	18.6468	0.0000
CD	0.14302	0.0276124	5.17959	0.0000

Analysis of Variance					
Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
Model	38.389	5	7.67781	314.69	0.0000
Residual	0.63436	26	0.02440		
Total(Corr.)	39.0234	31			

R-squared = 98.37%      R-squared (adjusted for d.f.) = 98.06%

The cube plot in Figure 6.6a shows the estimated log(Feeder setup) at selected combinations of USG stdev., Sequence, and Assignment with the other factors being held constant. It shows that the lowest log(Feeder setup) value, 4.74, occurs at the combination of high USG stdev., IIGS, and optimal feeder assignment. The main effect plot in Figure 6.6b shows that the IIGS and KTNS policy has the most significant effect which reduces  $(e^{6.34} - e^{4.93})/e^{6.34}$ , i.e. 75.6%, of feeder setups from random sequence case. The component to feeder assignment solution alone contributes about 40% of feeder setup reduction. Notice that the difference of feeder setups between productions with low and high variation USG is 70%. The interaction AC in Figure 6.6c shows that the sequencing methodology would reduce significant feeder setups (91%) in a production of uniform component usage compared to 32% of reduction in a highly distributed usage production.



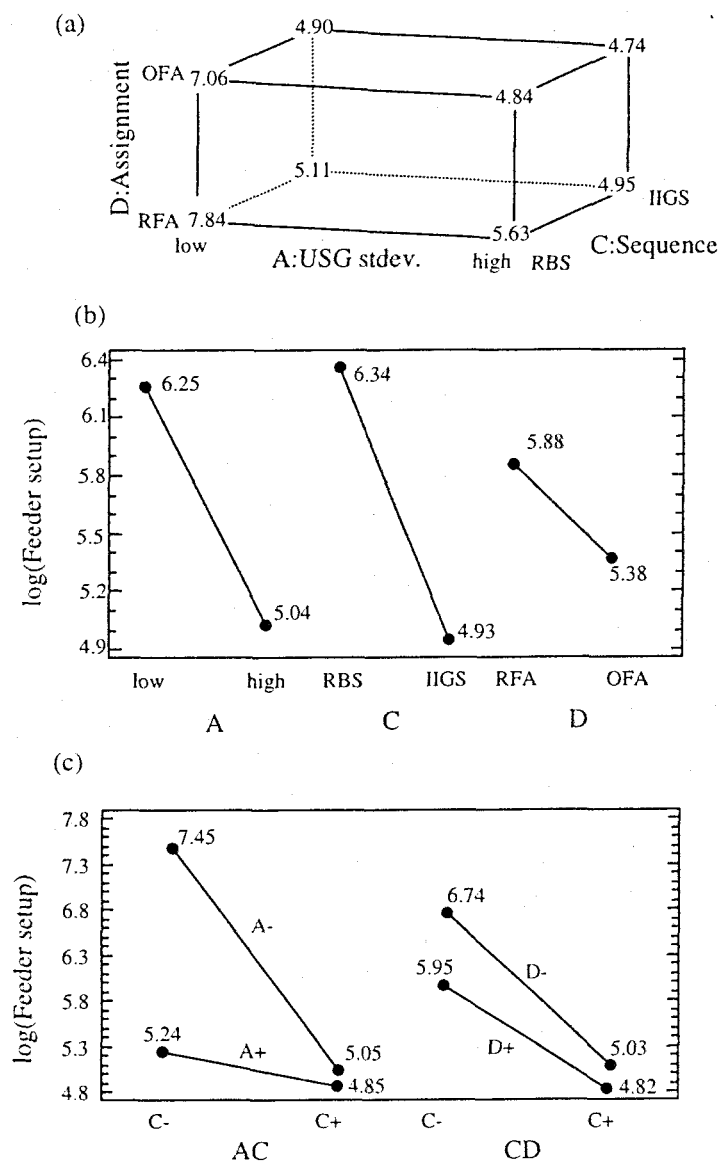


Figure 6.6 a) Cube plot, b) main effect plot, and c) interaction plot of Assignment, USG stdev., and Sequence for log(Feeder setup).

### 6.3.3 Analysis of Effects on Solution Times

The solution time is analyzed using the same procedure as feeder setups. Appendix C.4 lists the ANOVA for CPU times, and shows several significant two and three way interactions. The ANOVA of reduced model in Appendix C.5 drops all the insignificant effects in the full model and shows the interactions that are still significant.

The model explains 95.8% of data variability and can not be further reduced. The backward selection stepwise regression displayed in Appendix C.6 identifies the significant effects again at 99% confidence level with the final fitted regression model shown in Table 6.5. The main effect and interaction plots are displayed in Figure 6.7.

Table 6.5 Backward selection stepwise regression for CPU times.

Parameter	Estimate	Standard Error	T Statistic	P-Value
CONSTANT	143.906	2.04345	70.4231	0.0000
A:USG stdev.	16.2188	2.04345	7.93694	0.0000
B:REQ stdev.	9.78125	2.04345	4.78663	0.0001
C:Sequence	30.2813	2.04345	14.8187	0.0000
D:Assignment	-14.7188	2.04345	-7.20289	0.0000
AC	-28.0313	2.04345	-13.7176	0.0000
AD	6.59375	2.04345	3.22677	0.0039
BD	6.40625	2.04345	3.13502	0.0048
ABC	-8.28125	2.04345	-4.05258	0.0005
ACD	-11.6563	2.04345	-5.70420	0.0000

#### Analysis of Variance

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
Model	82145.0	9	9127.23	68.31	0.0000
Residual	2939.69	22	133.622		
Total(Corr.)	85084.7	31			

R-squared = 96.545 percent

R-squared (adjusted for d.f.) = 95.1316 percent

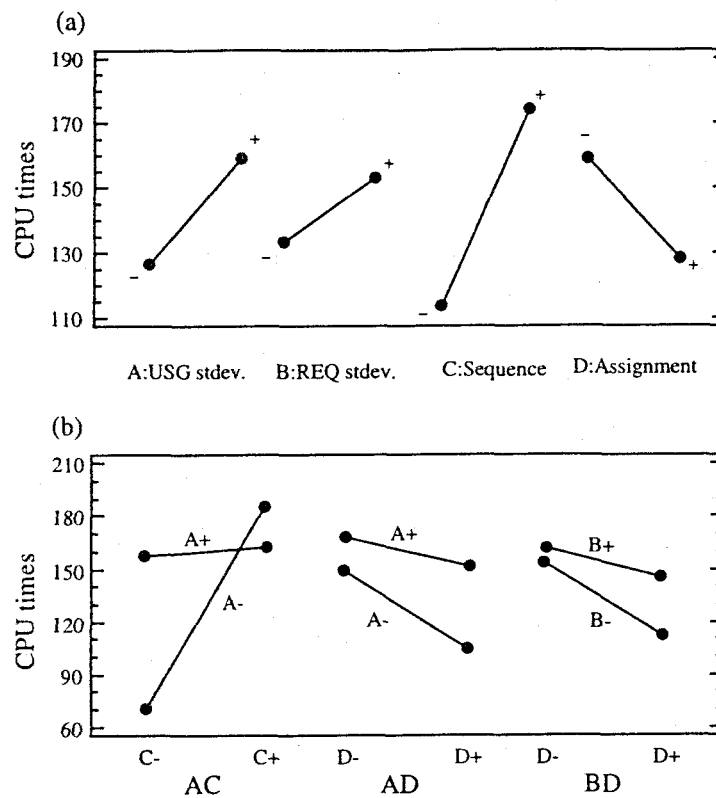


Figure 6.7 a) main effect plot, b) interaction plot of all factors for CPU times.

It can be concluded from the results that the algorithm efficiency is affected by the main effects and their interactions. Figure 6.7a shows that the developed algorithms are more efficient in productions with low usage and requirement variations. Application of IIGS and KTNS policies requires more search loops and, consequently, more computer time. Notice that the assignment effect plays an important role in facilitating an algorithm's efficiency. An optimal or best component to multi-track feeder assignment can speed up setup planning about 20%. Figure 6.7b shows AC to be an important interaction effect on solution times. The random PCB sequence in a uniform component usage production (A-, C-) can be easily solved; however, the resulting solution is the worst (see Figure 6.6c). The other two interactions, AD and BD, basically follow the main effect D that planned

assignment will make the setup planning more efficient. The three way interactions ABC and ACD are display in cube plots in Figure 6.8 with the fourth factor held at its center value.

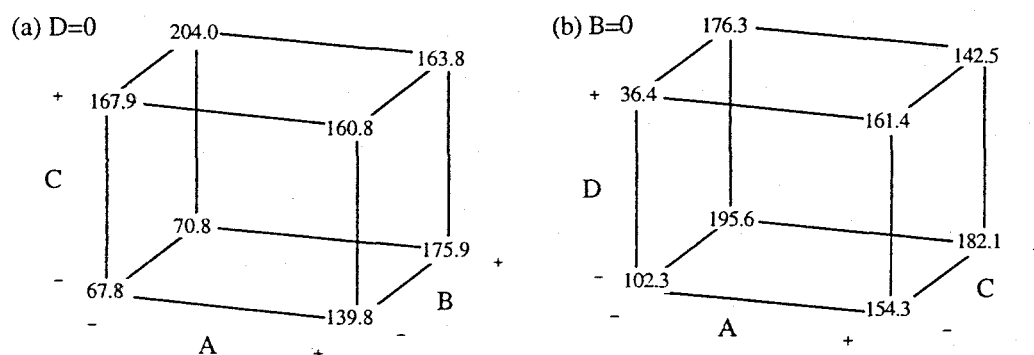


Figure 6.8 Cube plot of interaction a) ABC, and b) ACD for CPU times

Figure 6.8a shows that the CPU times are the lowest at (A-, C-, B-) and the highest at (A-, C+, B+). Figure 6.8b indicates that the lowest CPU time occurs at (A-, C-, D+) and the highest at (A-, C+, D-). Notice that these results are consistent with the two-way interaction AC. In the KTNS policy, the components to be kept on the machine are chosen in the order of requirements, needed soonest, and early installation to fill in the empty slots. It requires more time if there are many search loops that need to be explored for all three stages. The reason why the CPU times of random sequence (C-) and assignments with high similarity values (D+) for ELH and ELL (A-) are lower than EHH and EHL cases is that the component to be kept can be more easily found if the component is commonly used in different PCBs. Therefore most of search loops may finish before stage 2. In contrast, a large portion of search loops with IIGS sequence (C+) cases go through all stages and require more search times.

## 6.4 Evaluation of Algorithm Combinations

The simulation data sets are also used to investigate the performance of different combinations of algorithms. The PCB sequence obtained by IIGS is compared to random sequence. The component-to-feeder assignment methods include:

1. Assignment with minimum similarity value (MIN)
2. Random assignment (RFA)
3. Usage-based assignment (USG)
4. MinMax (MMX)
5. SWAP with random feeder assignment (SWRFA)
6. SWAP with usage-based assignment (SWUSG)
7. SWAP with MinMax assignment (SWMMX)
8. Integrated assignment (INTEG)

The assignment results for the simulation data are summarized in Table 6.6. The SWAP and INTEG heuristics find most of the best solutions. The similarity value by usage-based assignment decreases along with the variations of component usage and board requirement. The results for both sets of data are similar and the results of the second set of data are illustrated in Figures 6.9 and 6.10.

Table 6.6 Similarity values of each assignment

Data set	MIN	RFA	USG	MMX	SWRFA	SWUSG	SWMMX	INTEG
C	0	876	3522	6402	6394	6342	6448	6516*
EMM1	0	1054	8622	9218	9308*	9304	9274	9258
EHH1	0	1714	8994	9230	9252	9274*	9254	9258
EHL1	0	896	8890	9366	9414*	9404	9414*	9410
ELH1	0	1552	3492	9274	9316*	9300	9302	9308
ELL1	0	528	1486	9472*	9472*	9472*	9472*	9472*
EMM2	0	906	8708	9178	9212	9298*	9278	9234
EHH2	0	1502	8972	9278	9260	9298	9298	9302*
EHL2	0	650	8856	9282	9296	9296	9294	9298*
ELH2	0	1328	3502	9240	9292	9286	9302*	9284
ELL2	0	592	1558	9414	9416*	9416*	9416*	9416*

\*: best solution

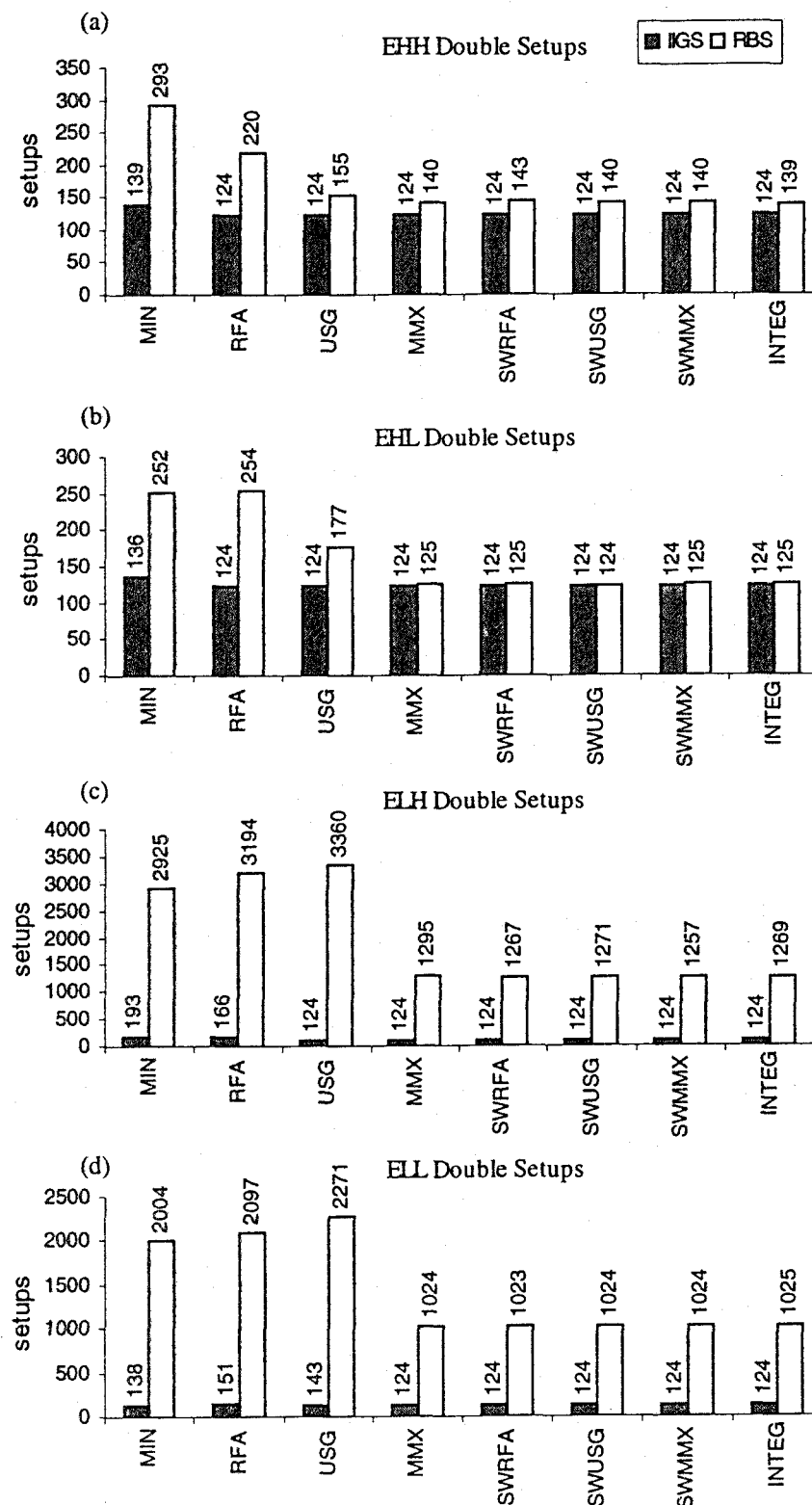


Figure 6.9 Evaluation results of feeder setup for data set 2

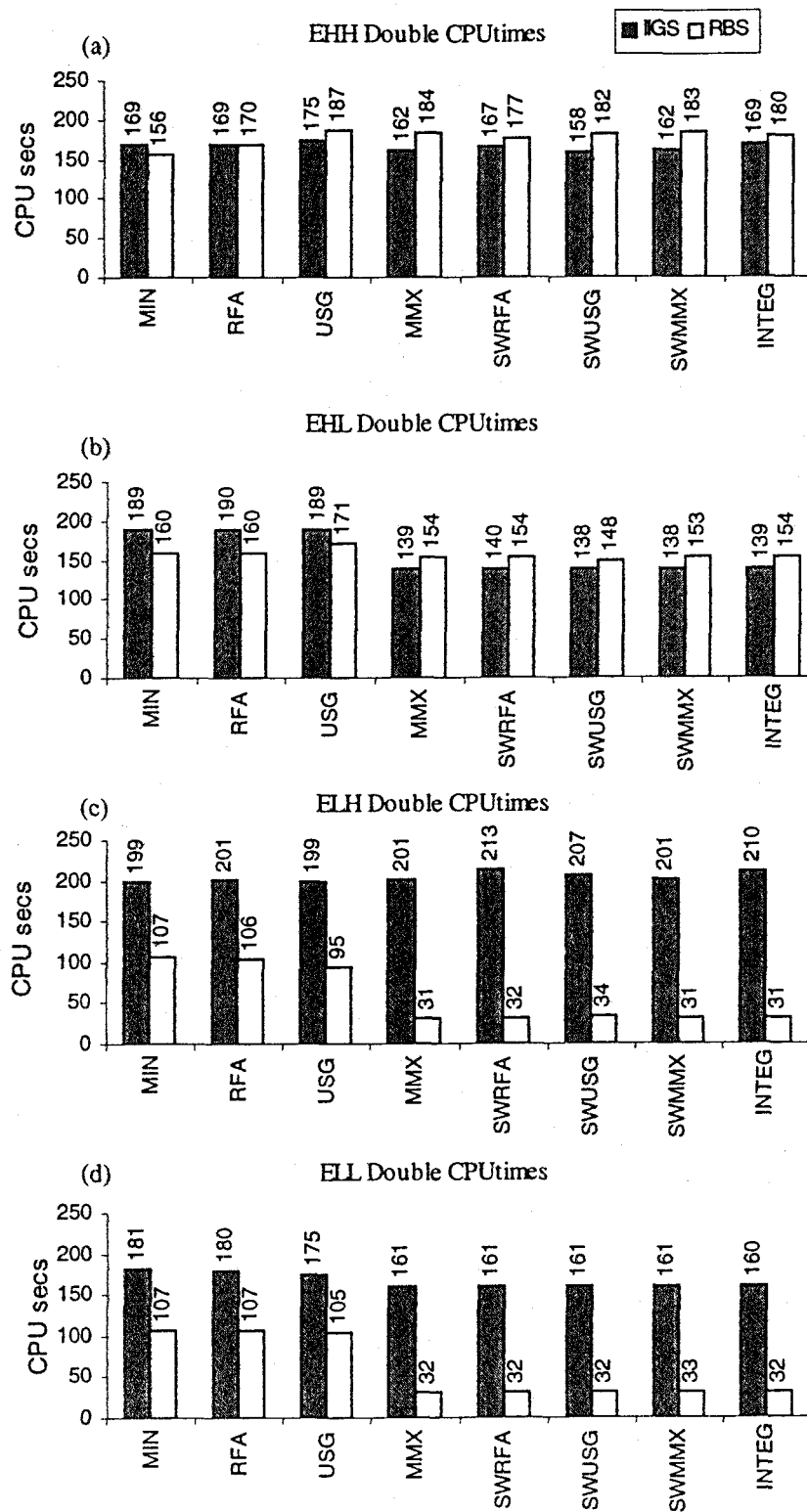


Figure 6.10 Evaluation results of CPU times for data set 2

The analyses of results are summarized as follows:

1. Assignments with high similarity values (MMX, SWRFA, SWUSG, SWMMX, and INTEG) all have the same, or very close, number of feeder setups. The best assignment solutions usually have the minimal feeder setups but not for every case.
2. The usage-based assignment is the worst choice in ELH and ELL cases if PCB boards are randomly sequenced. Since all components are used in similar frequency, assigning components with the same or close usage frequency is not better (in fact, may be worse) than a random assignment.
3. As shown by the experiment results, the effect of the IIGS and KTNS policy significantly reduces the feeder setups in ELH and ELL cases (Figure 6.9c and 6.9d), and is robust for different production environments. The differences of feeder setups between IIGS and random sequence are close in EHH and EHL cases with assignments of high similarity values.
4. The usage-based (USG) and SWAP with USG initial solution (SWUSG) assignment methods yield close results for both planned and random PCB sequences in EHH and EHL cases (Figure 6.9a and 6.9b). However, the differences are immense in ELH and ELL cases (Figure 6.9c and 6.9d). Both methods are currently used in industry.
5. The results of the industry data C are shown in Figure 6.11. Compared with Figure 6.9 and 6.10, the pattern of the results is similar between EHH and EHL cases. A review of Table 6.2 shows that the USG stdev. of Data C is close to EHH and EHL, and the REQ stdev. is between EHH and EHL. This proves that the simulation data works well.



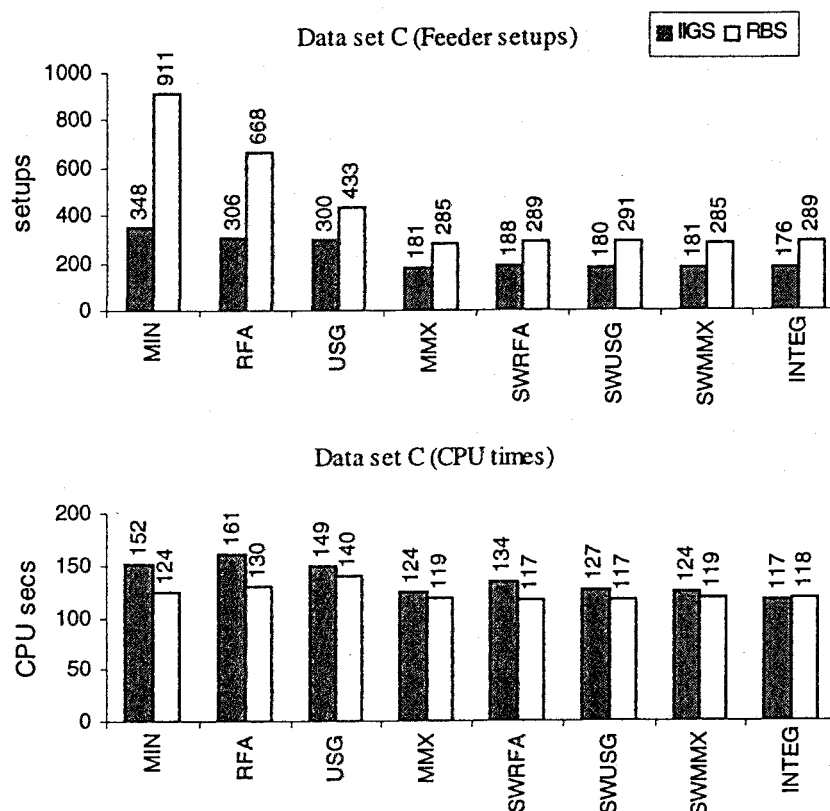


Figure 6.11 Evaluation results for industry data set C

## 6.5 Chapter Summary and Conclusions

In this chapter, four extreme cases are created using a random number generator to simulate different production types. A full factorial experiment with replicates is designed and the backward selection stepwise regression method is used to analyze the main and interaction effects. A full investigation of combinations of developed heuristics is carried out to verify the analysis.

The following conclusions can be drawn from this chapter:

1. The combination of assignment and sequencing methodology is a robust solution for different production environments.

2. The usage-based, or SWAP/USG, assignment works well in a high variation USG production even without proper PCB sequence. However, if the components are used uniformly by different PCBs, the usage-based assignment becomes the worst choice.
3. The effect of the developed sequencing method and feeder setup planning policy is more significant than the assignment method. This indicates that the PCB sequence is more important than the component to multi-track feeder assignment.
4. The efficiency of the application is affected by the variations of USG and REQ. However, the problem size effect is more significant than USG and REQ variations.

## CHAPTER 7 SUMMARY AND CONCLUSIONS

This final chapter summarizes the research work and its contributions. The limitations of the developed methodology are also discussed. Finally, some potential topics for extension of this research are suggested.

### 7.1 Summary of the Dissertation

The multi-track feeder setup management becomes a challenging issue in a low-volume, high-mix production environment. The benefit of increasing the capacity of SMT machines is diminished if components are not properly assigned to multi-track feeders. The feeder setup problem is analyzed and decomposed into two sub-problems: the component to multi-track feeder assignment problem and the PCB sequencing problem. However, the interrelationship of these two sub-problems shows that no optimal solutions exist separately.

A detailed review of published literature helped identify characteristics of the two sub-problems. Each sub-problem belongs to the *NP*-complete class, which suggests the difficulty for solving such problems increase as problem size grows.

An optimization model for assignment problem is formulated with applications on a commercial mixed integer-programming solver. Several heuristic algorithms are also developed to overcome the limitations and difficulties of applying the integer-programming model. Data sets from literature and industry combined with the randomly generated data are used to verify, analyze, and evaluate the performance and solution quality of the heuristics.

The PCB sequencing sub-problem is solved in three stages: component and PCB grouping, intra-group and inter-group sequencing, and feeder setup planning. Methods from published literatures are adapted to solve this problem. The devel-

oped methodologies for each sub-problem are then integrated for the final solution. Again, data sets from literature and industry are used to verify and evaluate the developed methods.

Finally, the industrial data are carefully analyzed for factors that may cause performance variation in use of the developed methods in different production environments. A random number generator is developed to simulate characteristics of industrial data and to generate experimental data sets with more variability and variety. Experimentation is designed and carried out to investigate the effects of pertinent factors on simulation results.

## 7.2 Conclusions

Multi-track feeder setup reduction for a low volume, high mix production environment involves consideration of component-to-feeder assignment and PCB sequencing problems. Interrelationship between these two problems is investigated and described in this research along with the following conclusions:

1. The component to multi-track feeder assignment is identified as a multi-dimension symmetric assignment, and can be formulated as an integer programming model with the objective of maximizing the total sum of similarity values of the assignment. The application of the optimization model may be limited with large-sized problems due to computational resources.
2. The construction of similarity matrix is an obstacle to solve large-sized problems. It is more difficult to obtain and handle the similarity matrix for multi-dimension symmetric assignment. Most of the CPU time and memory are consumed at this stage.

3. The greedy search algorithm can be used for solving the multi-dimension symmetric assignment problem. The solution quality is the lowest compared to the other heuristics and the efficiency is affected by size of similarity matrix.
4. The neighborhood search with an initial solution set is also designed to solve multi-dimension symmetric assignment problems. The solution quality depends on the initial solution, and the efficiency is dependent of the number of feeder tracks.
5. The Hungarian algorithm, originally used to solve asymmetric assignment problem, can be utilized to reduce the initial problem size for double-feeder assignment (two-dimension symmetric assignment). The performance of the Hungarian algorithm is significantly affected by the distribution of the similarity matrix.
6. For distributed component usage frequency cases, the number of multi-track feeder setups decreases as the similarity value of assignment increases. For uniform component usage cases, the usage-based assignment method is worst if PCBs are randomly sequenced.
7. PCB sequence has a more significant effect on multi-track feeder setup reduction than the component to feeder assignment, especially when the component usage frequency is uniform.
8. It can reduce 85% of feeder setups for a production that updates from single feeder to double feeder system with the implementation of the developed methodology.
9. The efficiency of the developed methodology is affected by the variations of component usage and PCB requirement.

The methodologies developed in this research can be applied in different areas. Symmetric assignment is a commonly encountered problem. There is still a need for efficient algorithm to solve large-sized problems. The largest problem size tested in this research is  $2000 \times 2000$ , which requires 1,999,000 non-zero decision variables and 2000 constraints for the IP model. The solution times are within 70 seconds and the solution quality is within 0.01% to the upper bound value. Although the proposed methods are prone to be affected by the distribution of similarity matrix, it is still efficient for problems within the size range evaluated.

The limitations for this research application can be summarized as follows:

1. The problem size reduction procedure implemented in the integrated assignment heuristic is limited to the double-feeder case because the Hungarian algorithm can only solve two dimensional assignment problems.
2. This research is developed based on single machine case with unlimited feeder and component resource. Thus, it can not be directly applied on a multiple assembly lines problem or limited resource case.

### **7.3 Suggestions for Future Research**

This research is the first study focused on the relationship between multi-track feeder assignment and PCB sequencing. Some assumptions made in this research can be relaxed for other situations. The objective of this research is to minimize the feeder setup in a low volume and high mixed production; this can be altered for other considerations. The following suggestions provide some possibilities and directions for future research:

### **7.3.1 *Integration with Component Placement Sequence Problem***

The sequence of component placement is known to be a major contributor to processing time. In a production that requires consideration of process times, in addition to setup times, the component to multi-track feeder assignment used here may create a potential conflict to both objectives because the adjacent placement components could be assigned to two feeders far apart.

Feeder or board latency is not considered in this research since process time is not as critical as setup time in a low volume, high mix production environment. Components are assigned only according to their similarity values. If the component placement sequences can be determined in advance of batch production, then the components can be assigned by their similarity values and the frequency of adjacent sequencing. However, the biggest challenge encountered in this new situation may be the identification of placement sequences.

Another suggestion to solve the component placement sequence problem is to develop placement sequences before applying grouping procedures. Components that can be sequenced adjacently are grouped in families and assigned to feeders according to their similarity values within families. The placement sequences are then planned based on this assignment and the sequence families.

### **7.3.2 *Integration with Feeder Allocation Problem***

As mentioned in chapter 3, the process time can be reduced if components (in single-feeder system) are allocated as close as possible to minimize the total distance of feeder movement. This requires reallocation of feeders between two assemblies, which may be in conflict with the objective of minimizing setups by keeping the feeders in their current locations.

The allocation of multi-track feeders to carriage slots is more complicated than in single feeder systems because components are already assigned to feeders and fixed. If this problem can be integrated with the extension suggested in the previous section and the two problems can be solved simultaneously, the resulting solution may be more effective.

### *7.3.3 Application in Limited Resource Case*

In the limited resource case, some rarely used components or special feeders are shared among different machines or assembly lines. Once these components are not needed in the next assembly, they are unloaded and installed on different machines. The frequently used and high inventory components can be kept on the machines until they are no longer used. Therefore, the slot spaces may at times be not fully utilized due to limited feeders or components.

The PCB sequencing methodology developed in this research assumes that the carriage slots are always occupied and the components or feeders can be left on the machine even they are not needed. The KTNS policy and the modified version for multi-track feeder system are also based on the same assumption.

A suggestion for the limited resource case is to group the components or feeders by their usage or inventory levels, and then determine whether they should be shared among machines or not. In this way, some frequently used feeders and components are fixed on the machines and some are configurable and shared among machines. It will require more effort to adapt the methodologies developed in this research to the limited resource case because both assignment and feeder setup planning methods need to be modified.



#### **7.3.4 Integration with Workload Balancing Problem**

For a production with multiple assembly lines, workload balancing is important and necessary to eliminate production bottlenecks and maximize throughput. Assessment of process times and setup times for each PCB batch are needed before the balancing process, which is also a *NP*-complete problem.

The methodologies developed in this research are all based on the single machine case. This research can not be directly applied to the line balancing case since different combination of PCBs results in different sequences, process times, and setup times. By involving multi-track feeders, the dynamic characteristic of the line balancing problem becomes more complicated.

## REFERENCES

- [1] Ammons J. C., Carlyle M., Cranmer L., Depuy G., Ellis K., McGinnis L. F., Tovey C.A. and Xu H., "Component Allocation To Balance Workload In Printed Circuit Card Assembly Systems", *IIE Transactions*, **29** (1997), 265-275.
- [2] Ang C. L., "On The Performance Of Hybrid Multi-Cell Flexible Manufacturing Systems", *International Journal of Production Research*, **33** (1995), No.10, 2779-2799.
- [3] Askin R. G., Dror M., Vakharia A. J., "Printed Circuit Board Family Grouping And Component Allocation For A Multimachine, Open-Shop Assembly Cell", *Naval Research Logistics*, **41** (1994), 587-608.
- [4] Axelson J., *Making Printed Circuit Boards*, McGraw-Hill, Inc., 1993.
- [5] Balinski M. L., Gomory R. E., "A Primal Method for the Assignment and Transportation Problems", *Management Science*, **10** (1964), No. 3, 572-593.
- [6] Ballakur A., Steudel H. J., "A Within-Cell Utilization Based Heuristic For Designing Cellular Manufacturing Systems", *International Journal of Production Research*, **25** (1987), No. 5, 639-665.
- [7] Basu A., Hyer N., Shtub A., "An Expert System Based Approach To Manufacturing Cell Design", *International Journal of Production Research*, **33** (1995), No. 10, 2739-2755.
- [8] Beasley J. E., "Linear programming on Cray supercomputers", *Journal of the Operational Research Society*, **41** (1990), 133-139, 1990.
- [9] Beasley J. E., OR-Library: <http://mscmga.ms.ic.ac.uk/info.html>
- [10] Ben-Arieh D., Dror M., "Part Assignment To Electronic Insertion Machines: Two Machine Case", *International Journal of Production Research*, **28** (1990), No. 7, 1317-1327.
- [11] Bhaskar G., Narendran T. T., "Grouping PCBs For Set-Up Reduction: A Maximum Spanning Tree Approach", *International Journal of Production Research*, **34** (1996), No. 3, 621-632.
- [12] Boctor F. F., "The Minimum-Cost, Machine-Part Cell Formation Problem", *International Journal of Production Research*, **34** (1996), No. 4, 1045-1063.

- [13] Capps C. H., *Setup Reduction in PCB Assembly: A Group Technology Application Using Genetic Algorithm*, Master thesis, Oregon State University, 1998.
- [14] Carmon T. F., Maimon O. Z., Dar-El E. M., "Group Set-Up For Printed Circuit Board Assembly", *International Journal of Production Research*, **27** (1989), No. 10, 1795-1810.
- [15] Carpaneto, G., Toth P., "Algorithm 548 Solution of the Assignment Problem", *ACM Transactions on Mathematical Software*, **6** (1980), No. 1, 104-111.
- [16] Chang T. C., Terwilliger Jr. J., "A Rule Based System For Printed Wiring Assembly Process Planning", *International Journal of Production Research*, **25** (1987), No. 10, 1465-1482.
- [17] Charles-Owaba O. E., Lambert B. K., "Sequence Dependent Machine Set-Up Times And Similarity Of Parts: A Mathematical Model", *IIE Transactions*, **20** (1988), No. 1, 12-21.
- [18] Chen F. F., Ker J. I., Kleawpatinon K., "An Effective Part-Selection Model For Production Planning Of Flexible Manufacturing Systems", *International Journal of Production Research*, **33** (1995), No. 10, 2617-2683.
- [19] Chen Y. J., Askin R. G., "A Multiobjective Evaluation Of Flexible Manufacturing System Loading Heuristics", *International Journal of Production Research*, **28** (1990), No. 5, 895-911.
- [20] Cheng C. H., Madan M. S., Motwani J., "Designing Cellular Manufacturing Systems By A Truncated Tree Search", *International Journal of Production Research*, **34** (1996), No. 2, 349-361.
- [21] Cunningham P., Browne J., "A LISP-Based Heuristic Scheduler For Automatic Insertion In Electronics Assembly", *International Journal of Production Research*, **24** (1986), No. 6, 1395-1408.
- [22] Daskin M. S., Maimon O., Shtub A., and Braha D., "Grouping Components In Printed Circuit Board Assembly With Limited Component Staging Capacity And Single Card Setup: Problem Characteristics And Solution Procedures", *International Journal of Production Research*, **35** (1997), No. 6, 1617-1638.
- [23] Depuy G. W., *Component Allocation To Balance Workload In Printed Circuit Card Assembly Systems*, Doctoral dissertation, Georgia Institute of Technology, 1995.

- [24] Dessouky M. M., Adiga S. and Park K., "Design And Scheduling Of Flexible Assembly Lines For Printed Circuit Boards", *International Journal of Production Research*, **33** (1995), No. 3, 757-775.
- [25] Devine M. D., "A Model for Minimizing the Cost of Drilling Dual Completion Oil Wells", *Management Science*, **20** (1973), No. 4, 532-539.
- [26] Ellis K. P., *Analysis Of Setup Management Strategies In Electronic Assembly Systems*, Doctoral dissertation, Georgia Institute of Technology, 1996.
- [27] Fathi Y., Taheri J., "A Mathematical Model for Loading the Sequencers in a Printed Circuit Pack Manufacturing Environment", *International Journal of Production Research*, **27** (1989), No. 8, 1305-1316.
- [28] Foulds L. R., *Combinatorial Optimization for Undergraduates*, Springer-Verlog, 1984.
- [29] Fourer R., Gay D. M., Kernighan B. W., *AMPL: A Modeling Language For Mathematical Programming*, Duxbury Press, 1993
- [30] Garey, M. R., and Johnson D. S., *Computers And Intractability: A Guide To The Theory Of NP-Completeness*, W. H. Freeman, San Francisco, 1979.
- [31] Gupta Y., Gupta M., Kumar A., Sundaram C., "A Genetic Algorithm-Based Approach To Cell Composition And Layout Design Problems", *International Journal of Production Research*, **34** (1996), No. 2, 447-482.
- [32] Hashiba S., Chang T. C., "PCB Assembly Setup Reduction Using Group Technology", *Computers & Industrial Engineering*, **21** (1991), No. 1-4, pp. 453-457.
- [33] Hillier M. S., *Models For Manufacturing System Design: Operation Assignment And Product Grouping (Hewlett Packard)*, Doctoral dissertation, Stanford University, 1994.
- [34] Huang Yu-Wen, Srihari K., Adriance J., Westby G., "A Solution Methodology For The Multiple Batch Surface Mount PCB Placement Sequence Problem", *Advances in Electronic Packaging EEP ASME*, **4-1** (1993), 373-379.
- [35] Ji P., Lee W. B. And Li H., "A New Algorithm For The Assignment Problem: An Alternative To The Hungarian Method", *Computers & Operation Research*, **24** (1997), No. 11, 1017-1023.

- [36] Ji Z., Leu M. C., Wong H., "Development And Implementation Of Linear Assignment Algorithm For Assembly Of PCB Components", *Advances in Electronic Packaging EEP ASME*, 4-1 (1993), 365-371.
- [37] Jiang J. J., *A Shop Floor Scheduling System For Repetitive Flexible Flow Lines (Scheduling)*, Doctoral dissertation, Arizona State University, 1994.
- [38] Johri P. K., "Engineering A Circuit Board Assembly Line For A Desired Capacity And Flowtime", *Journal of Manufacturing Systems*, 10 (1990), No. 6, 492-500.
- [39] Kamal S., Burke L. I., "Fact: A New Neural Network-Based Clustering Algorithm For Group Technology", *International Journal of Production Research*, 34 (1996), No. 4, 919-946.
- [40] Karp R. M., "Reducibility Among Combinatorial Problems" in R. E. Miller and J. W. Thatcher (Eds.), *Complexity of Computer Computations*, Plenum Press, 1972.
- [41] King J. R., "Machine-Component Grouping In Production Flow Analysis: An Approach Using A Rank Order Clustering Algorithm", *International Journal of Production Research*, 18 (1980), No. 2, 213-232.
- [42] Kuhn H. W., "The Hungarian Method for the Assignment Problem", *Naval Res. Logist. Quart.*, 2 (1955), 83-97.
- [43] Kung H. K., Changchit C., "A Just-In-Time Simulation Model Of A PCB Assembly Line", *Computers & Industrial Engineering*, 20 (1991), No. 1, 17-26.
- [44] Kusiak A., "The Generalized Group Technology Concept", *International Journal of Production Research*, 25 (1987), No. 4, 561-569.
- [45] Lach A. A., *Printed Circuit Board Assembly – Impact Of Technological And Market Trends On Process Planning And Optimization With Emphasis On Component Ordering Optimization Problems*, Doctoral dissertation, Northwestern University, 1994.
- [46] Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Rinehart and Winston, 1976
- [47] Leipala T., Nevalainen O., "Optimization Of The Movements Of A Component Placement Machine", *European Journal of Operational Research*, 38 (1989), 167-177.

- [48] Maimon O., Shtub A., "Grouping Methods For Printed Circuit Board Assembly", *International Journal of Production Research*, **29** (1991), No. 7, 1379-1390.
- [49] Maimon O. Z., Braha D., "A Genetic Algorithm Approach To Scheduling PCBs On A Single Machine", *International Journal of Production Research*, **36** (1998), No. 3, 761-784.
- [50] McAuley J., "Machine Grouping for Efficient Production", *The Production Engineering*, **51** (1972), 53-57.
- [51] McCormick W., Schweitzer P. and Whit T., "Problem Decomposition and Data Reorganization by a Clustering Technique", *Operations Research*, **20** (1972), 993-1008.
- [52] McGinnis L. F., Ammons J. C., Carlyle M., Cranmer L., Depuy G. W., Ellis K. P., Tovey C. A., Xu H., "Automated Process Planning For Printed Circuit Card Assembly", *IIE Transactions*, **24** (1992), No. 4, 18-30.
- [53] Montgomery D. C., *Design and Analysis of Experiments*, 3rd Edition, John Wiley & Sons, 1991
- [54] Mosier C. T., "An Experiment Investigating The Application Of Clustering Procedures And Similarity Coefficients To The GT Machine Cell Formation Problem", *International Journal of Production Research*, **27** (1989), No. 10, 1811-1835.
- [55] Moyer L. K., Gupta S. M., "SMT Feeder Slot Assignment for Predetermined Component Placement Paths", *Journal of Electronics Manufacturing*, **6** (1996), No. 3, 173-192.
- [56] Moyer L. K., Gupta S. M., "Simultaneous Component Sequencing and Feeder Assignment for High Speed Chip Shooter Machines", *Journal of Electronics Manufacturing*, **6** (1996), No. 4, 271-305.
- [57] Moyer L. K., Gupta S. M., "An Efficient Assembly Sequencing Heuristic for Printed Circuit Board Configurations", *Journal of Electronics Manufacturing*, **7** (1997), No. 2, 143-160.
- [58] Murty K. G., *Linear And Combinatorial Programming*, Wiley, New York, 1976.
- [59] Nemhauser G. L., Wolsey L. A., *Integer and Combinatorial Optimization*, Wiley-Interscience Publication, 1988.

- [60] Noble P J. W., *Printed Circuit Board Assembly*, Halsted Press, 1989
- [61] Peters B. A., Subramanian G. S., "Analysis Of Partial Setup Strategies For Solving The Operational Planning Problem In Parallel Machine Electronic Assembly Systems", *International Journal of Production Research*, **34** (1996), No. 4, 999-1021.
- [62] Prasad R. P., *Surface Mount Technology*, second edition, Chapman & Hall, 1997.
- [63] Randhawa S. U., McDowell E. D., Faruqui S., "An Integer Programming Application to Solve Sequencer Mix problems in Printed Circuit Board Production", *International Journal of Production Research*, **23** (1985), No. 3, 543-552.
- [64] Rowland R., Belangia P., *Applied Surface Mount Assembly, A Guide To Surface Mount Materials And Processes*, Van Nostrand Reinhold, 1993
- [65] Sanders R. C., *Operational Planning For Electronic Assembly On A Two-Machine Mixed-Model Assembly Line*, Doctoral dissertation, Texas A&M University, 1996.
- [66] Sarin S. C., Erel E., "Development Of Cost Model For The Single-Model Stochastic Assembly Line Balancing Problem", *International Journal of Production Research*, **28** (1990), No. 7, 1305-1316.
- [67] Shargal M., Shekhar S., Irani S. A., "Evaluation Of Search Algorithms And Clustering Efficiency Measures For Machine-Part Matrix Clustering", *IIE Transactions*, **27** (1995), pp. 43-59.
- [68] Shtub A., Maimon O., "Role Of Similarity Measures In PCB Grouping Procedures", *International Journal of Production Research*, **30** (1992), No. 5, 973-983.
- [69] Sneath P. H. A., Sokal R. R., *Principles of Numerical Taxonomy*, Freeman & Co., 1968.
- [70] Strauss R., *Surface Mount Technology*, Butterworth-Heinemann Ltd, Oxford, 1994
- [71] Sule D. R., "A Systematic Approach For Machine Grouping In Cellular Manufacturing", *International Industrial Engineering Conference & Proceedings*, 1989, 619-624.

- [72] Sule D. R., "A Heuristic Procedure For Component Scheduling In Printed Circuit Pack Sequencers", *International Journal of Production Research*, **30** (1992), No. 5, 1191-1208.
- [73] Suresh G., Vinod V. V., Sahu S., "A Genetic Algorithm For Facility Lay-out", *International Journal of Production Research*, **33** (1995), No. 12, 3411-3423.
- [74] Takvorian A., *Topics In Combinatorial Optimization: Graph Planarization And Product Sequencing In Assembly Lines*, Doctoral dissertation, The University of Texas at Austin, 1994.
- [75] Tang C. S., Denardo E. V., "Models Arising From A Flexible Manufacturing Machine, Part I: Minimization Of The Number Of Tool Switches", *Operations Research*, **36** (1988), No. 5, 767-777.
- [76] Tang C. S., Denardo E. V., "Models Arising From A Flexible Manufacturing Machine, Part II: Minimization Of The Number Of Switching Instants", *Operations Research*, **36** (1988), No. 5, 778-784.
- [77] Towle K. A., "Line Configuration For Today's Highly Complex Manufacturing", *SMT Magazine*, November 1998, 70-75.
- [78] Verma P., Ding F. Y., "A Sequence-Based Materials Flow Procedure For Designing Manufacturing Cells", *International Journal of Production Research*, **33** (1995), No. 12, pp. 3267-3281.
- [79] Wright M. B., "Speeding Up The Hungarian Algorithm", *Computers Operations Research*, **17** (1990), No. 1, 95-96.
- [80] Xu Z., Li Y., Randhawa S., Carlson K., Kurschner R., "An Integrated Methodology for Surface Mount PCB Configuration", to appear in *Journal of Electronics Manufacturing*, 1998.
- [81] Zhu Z., Heady R. B., Reiners S., "An Efficient Zero-One Formulation Of The Cell Formation Problem", *Computers & Industrial Engineering*, **28** (1995), No. 4, 911-916.



## APPENDICES

**APPENDIX A**

### A.1 AMPL Model for Double Feeder Assignment Problem

```

param N > 0; # number of parts
param P{i in 1..N-1 ,j in i+1..N};
var X{i in 1..N-1,j in i+1..N} binary; # the assignment of part i and j
minimize T:
sum {i in 1..N-1, j in i+1..N} P[i,j] * X[i,j];
subject to row{i in 1..N}:
    sum {j in 1..N:j<>i} (if j>i then X[i,j] else X[j,i]) = 1;

```

## A.2 AMPL Data for Double Feeder Assignment Problem

```
param N:=10;
param P: 1 2 3 4 5 6 7 8 9 10 :=
1      . -2 0 -2 -2 -2 -1 -2 -2 1
2      . . -1 -4 -3 -2 -3 -3 -3 1
3      . . . -1 0 0 -1 0 0 1
4      . . . . -2 -2 -4 -2 -2 1
5      . . . . . -1 -1 -2 -2 1
6      . . . . . . -2 -2 -2 1
7      . . . . . . . -1 -1 1
8      . . . . . . . . -3 1
9      . . . . . . . . . 1
10     . . . . . . . . . ;
```

### A.3 Triple Symmetric Similarity Matrix for Sample Data

$i$	$j$	$k$	$p_{ijk}$	$i$	$j$	$k$	$p_{ijk}$
1	2	3	0	2	5	9	2
1	2	4	1	2	6	7	1
1	2	5	2	2	6	8	2
1	2	6	1	2	6	9	2
1	2	7	0	2	7	8	1
1	2	8	2	2	7	9	1
1	2	9	2	2	8	9	3
1	3	4	0	3	4	5	0
1	3	5	0	3	4	6	0
1	3	6	0	3	4	7	1
1	3	7	0	3	4	8	0
1	3	8	0	3	4	9	0
1	3	9	0	3	5	6	0
1	4	5	1	3	5	7	0
1	4	6	1	3	5	8	0
1	4	7	1	3	5	9	0
1	4	8	1	3	6	7	0
1	4	9	1	3	6	8	0
1	5	6	1	3	6	9	0
1	5	7	0	3	7	8	0
1	5	8	2	3	7	9	0
1	5	9	2	3	8	9	0
1	6	7	1	4	5	6	0
1	6	8	1	4	5	7	1
1	6	9	1	4	5	8	1
1	7	8	0	4	5	9	1
1	7	9	0	4	6	7	2
1	8	9	2	4	6	8	1
2	3	4	1	4	6	9	1
2	3	5	0	4	7	8	1
2	3	6	0	4	7	9	1
2	3	7	1	4	8	9	2
2	3	8	0	5	6	7	0
2	3	9	0	5	6	8	1
2	4	5	2	5	6	9	1
2	4	6	1	5	7	8	0
2	4	7	3	5	7	9	0
2	4	8	2	5	8	9	2
2	4	9	2	6	7	8	1
2	5	6	1	6	7	9	1
2	5	7	1	6	8	9	2
2	5	8	2	7	8	9	1

#### A.4 AMPL Model for Triple Feeder Assignment Problem

```

param N > 0; # number of parts <=12 {N mod 3=0}
param P{i in 1..N-2 ,j in i+1..N-1, k in j+1..N};
var X{i in 1..N-2 ,j in i+1..N-1, k in j+1..N} binary; # of the
    assignment of part i and j and k

maximize T:
sum {i in 1..N-2,j in i+1..N-1,k in j+1..N} P[i,j,k]*X[i,j,k];

subject to row{i in 1..N}:
sum {j in 1..N,k in j+1..N: j<>i and k<>i}
    (if (i<j and j<k) then X[i,j,k] else
     (if (k<j and j<i) then X[k,j,i] else
      (if (j<i and i<k) then X[j,i,k] else
       (if (j<k and k<i) then X[j,k,i] else
        (if (k<i and i<j) then X[k,i,j] else X[i,k,j] ))))) = 1;

```

#### A.5 AMPL Data for Triple Feeder Assignment Problem

```

param N:=9;
param P:=
[1,*,*] 2 3 0 2 4 1 2 5 2 2 6 1 2 7 0 2 8 2 2 9 2 3 4 0 3 5 0
        3 6 0 3 7 0 3 8 0 3 9 0 4 5 1 4 6 1 4 7 1 4 8 1 4 9 1
        5 6 1 5 7 0 5 8 2 5 9 2 6 7 1 6 8 1 6 9 1 7 8 0 7 9 0
        8 9 2
[2,*,*] 3 4 1 3 5 0 3 6 0 3 7 1 3 8 0 3 9 0 4 5 2 4 6 1 4 7 3
        4 8 2 4 9 2 5 6 1 5 7 1 5 8 2 5 9 2 6 7 1 6 8 2 6 9 2
        7 8 1 7 9 1 8 9 3
[3,*,*] 4 5 0 4 6 0 4 7 1 4 8 0 4 9 0 5 6 0 5 7 0 5 8 0 5 9 0
        6 7 0 6 8 0 6 9 0 7 8 0 7 9 0 8 9 0
[4,*,*] 5 6 0 5 7 1 5 8 1 5 9 1 6 7 2 6 8 1 6 9 1 7 8 1 7 9 1
        8 9 2
[5,*,*] 6 7 0 6 8 1 6 9 1 7 8 0 7 9 0 8 9 2
[6,*,*] 7 8 1 7 9 1 8 9 2
[7,*,*] 8 9 1 ;

```

### A.6 Pseudo-codes for Min-Max heuristic

Let  $\left\{ \begin{array}{l} S = \{x_{(1)}, x_{(2)}, \dots, x_{(n/2)}\}$ , the symmetric solution set  
where  $x_{(i)} = x_{ij}$ ,  $i < j$ , and all  $x_{(i)}$  are mutually exclusive.  
 $P = \{p_{(1)}, p_{(2)}, \dots, p_{(n/2)}\}$ , the descending sorted similarity matrix  
where  $p_{(i)} = p_{ij}$ ,  $i < j$ ,  $p_{(i)} > 0$   
 $k$  = number of loops

```

1: initialize S;
2: initialize P;
3: initialize k;
4: for a = 1 to k
5:   initialize sumOfP(a)
6:   for b = 1 to n/2
7:     if P = Φ then exit for
8:     if b = 1 then
9:       p = ath element in P, subscript (i, j)
10:    else
11:      p = the first element in P, subscript (i, j)
12:    end if
13:    sumOfP(a) = sumOfP(a) + p(i,j)
14:    update P, eliminate all elements in P that have associated subscripts i, or j
15:  next b
16:  if sumOfP(a) > sumOfP(a-1) then
17:    bestLoop = a
18:  end if
19:  initialize sumOfP(a)
20:  initialize P
21: next a
22: for b = 1 to n/2
23:   if b = 1 then
24:     S(b) = bestLoopth element in P, whose subscript is (i, j)
25:   else
26:     S(b) = the first element in P, whose subscript is (i, j)
27:   end if
28:   update P, eliminate all elements in P that have associated subscripts i, or j
29: next b

```

### A.7 Pseudo-codes for SWAP heuristic

Let  $\left\{ \begin{array}{l} S = \{x_{(1)}, x_{(2)}, \dots, x_{(v)}\}, \text{ the initial symmetric solution set} \\ = \{p_{(1)}, p_{(2)}, \dots, p_{(v)}\} \\ = \{p_{(i_1, j_1)}, p_{(i_2, j_2)}, \dots, p_{(i_v, j_v)}\} \\ \text{for components } u1, v1, u2, v2 : \\ P_A = p_{u1v1} + p_{u2v2}, P_B = p_{u1v2} + p_{u2v1}, P_C = p_{u1u2} + p_{v1v2} \end{array} \right.$

- 1:  $i = 0$ ;
- 2:  $i = i + 1$ ;
- 3: if  $i = n / 2$  then Goto Step 23;
- 4:  $j = i + 1$ ;
- 5:  $u1, v1$  are components in  $p_{(i)}$ ;
- 6:  $u2, v2$  are components in  $p_{(j)}$ ;
- 7: if  $P_A < P_B$  OR  $P_A < P_C$  then
- 8:   if  $P_A < P_B$  and  $P_A > P_C$  then
- 9:     Reassignment:  $p_{(i)} = p_{u1v2}, p_{(j)} = p_{u2v1}$ ;
- 10:   elseif  $P_A > P_B$  and  $P_A < P_C$  then
- 11:     Reassignment:  $p_{(i)} = p_{u1u2}, p_{(j)} = p_{v1v2}$ ;
- 12:   else
- 13:     if  $P_B < P_C$  then
- 14:       Reassignment:  $p_{(i)} = p_{u1u2}, p_{(j)} = p_{v1v2}$ ;
- 15:     else
- 16:       Reassignment:  $p_{(i)} = p_{u1v2}, p_{(j)} = p_{u2v1}$ ;
- 17:     end if
- 18:   end if
- 19: end if
- 20:  $j = j + 1$ ;
- 21: if  $j = n / 2$  then Goto Step 2
- 22: Goto Step 5
- 23: STOP

### A.8 Hungarian Algorithm Example

from (4.34)

#### 1. Matrix reduction

Transformed original similarity matrix	Row minimum	Reduced matrix after subtracting row minimum from rows								
1 -2 0 -2 -2 -2 -1 -2 -2	-2	3	0	2	0	0	0	1	0	0
-2 1 -1 -4 -3 -2 -3 -3 -3	-4	2	5	3	0	1	2	1	1	1
0 -1 1 -1 0 0 -1 0 0	-1	1	0	2	0	1	1	0	1	1
-2 -4 -1 1 -2 -2 -4 -2 -2	-4	2	0	3	5	2	2	0	2	2
-2 -3 0 -2 1 -1 -1 -2 -2	-3	1	0	3	1	4	2	2	1	1
-2 -2 0 -2 -1 1 -2 -2 -2	-2	0	0	2	0	1	3	0	0	0
-1 -3 -1 -4 -1 -2 1 -1 -1	-4	3	1	3	0	3	2	5	3	3
-2 -3 0 -2 -2 -2 -1 1 -3	-3	1	0	3	1	1	1	2	4	0
-2 -3 0 -2 -2 -2 -1 -3 1	-3	1	0	3	1	1	1	2	0	4
Column minimum		0	0	2	0	0	0	0	0	0

#### 2. Find the initial solution and cross out lines to cover all zeros, and

Column $j$	1	2	3	4	5	6	7	8	9	Row label
Row $i=1$	3	0	0	0	0 <sup>③</sup>	0	1	0	0	7,0
2	2	5	1	0 <sup>①</sup>	1	2	1	1	1	1,0
3	1	0	0 <sup>⑤</sup>	0	1	1	0	1	1	4,2,0
4	2	0	1	5	2	2	0 <sup>⑥</sup>	2	2	2,1,0
5	1	0 <sup>②</sup>	1	1	4	2	2	1	1	1,0
6	0 <sup>④</sup>	0	0	0	1	3	0	0	0	7,0
7	3	1	1	0	3	2	5	3	3	1,0
8	1	0	1	1	1	1	2	4	0 <sup>⑦</sup>	2,1,0
9	1	0	1	1	1	1	2	0 <sup>⑦</sup>	4	2,1,0
Column label	1,0	7,0	3,1	5,0	1,0	1,0	3,2,0	3,1,0	3,1,0	



### A.8 Hungarian Algorithm Example (continued)

3. Redistribute the zeros and find an optimal solution

Column $j$	1	2	3	4	5	6	7	8	9	Row label
Row $i=1$	3	1	0	1	0	0④	2	0	0	5,0
2	1	5	0	0	0③	1	1	0	0	5,2
3	1	1	0①	1	1	1	1	1	1	1,0
4	1	0	0	5	1	1	0⑥	1	1	3,0
5	0	0⑤	0	1	3	1	2	0	0	5,2
6	0⑥	1	0	1	1	3	1	0	0	4,1
7	2	1	0	0⑦	2	1	5	2	2	2,1
8	1	1	1	2	1	1	3	4	0②	1,0
9	1	1	1	2	1	1	3	0⑤	4	1,0
Column label	2	2,1	7,0	2	2,1	1,0	1,0	5,0	5,0	

## A.9 Computation results for Data set 2 –R1

Size	Opt. Val. (HA)*	Best	HA**		MMX		SWAP/RFA		SWAP/MMX		INTEG		5-Runs***	
			value	time	value	time	value	time	value	time	value	time	value	time
18	1616	1616	1616	0.05	1540	0.03	1508	0.001	1616	0.01	1616	0.02	1616	0.01
50	4808	4808	3888	0.05	4700	0.05	4626	0.01	4748	0.06	4808	0.10	4808	0.054
100	9858	9850	8676	0.15	9594	0.37	9462	0.05	9704	0.39	9850	0.15	9850	0.212
200	19878	19878	17292	0.57	19598	2.77	19186	0.20	19748	2.81	19878	0.56	19878	0.596
400	39955	39954	30766	1.63	39604	23.91	39038	0.85	39718	24.14	39948	1.72	39954	2.754
600	59982	59982	29190	2.78	59460	80.05	58916	2.04	59778	80.95	59962	3.81	59982	5.224
800	79998	79998	26200	4.9	79374	189.47	78710	3.91	79794	190.96	79998	8.81	79998	12.726
1000	100000	100000	26600	7.61	99554	374.03	98526	6.27	99724	376.28	99996	14.5	100000	15.568
1200	120000	120000	30400	11.01	119386	641.92	118534	9.44	119830	645.2	120000	23.56	120000	30.83
1400	140000	140000	35000	14.55	139440	1026.4	138494	13.35	139762	1032.7	140000	33.27	140000	37.59
1600	160000	160000	44400	19.36	159452	1647.4	158476	17.59	159818	1621.7	159974	34.27	160000	53.644
1800	180000	180000	32600	24.55	179456	2201.7	178374	22.45	179802	2206.8	180000	57.4	180000	57.236
2000	200000	200000	123200	29.72	199404	3029.4	198336	28.09	199800	3032.5	199960	35.7	200000	65.178

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is not used in the Integrate heuristic,

Reduction procedure will be terminated if two consecutive loops are not improving

# A.10 Computation results for Data set 2 –R2

Size	Opt. Val. (HA)*	Best	HA**		MMX		SWAP/RFA		SWAP/MMX		INTEG		5-Runs***	
			value	time	value	time	value	time	value	time	value	time	value	time
18	3262	3262	3262	0.001	3194	0.02	3216	0.001	3250	0.01	3262	0.01	3262	0.01
50	9618	9560	6166	0.09	9388	0.05	9198	0.01	9394	0.06	9560	0.16	9560	0.176
100	19678	19678	19678	0.28	19072	0.37	18870	0.05	19550	0.4	19678	0.29	19678	0.294
200	39702	39702	39702	1.09	39262	2.76	38438	0.21	39424	2.84	39702	1.1	39702	1.12
400	79785	79780	69824	2.26	78786	23.9	77988	0.93	79374	24.18	79780	2.57	79780	3.296
600	119870	119870	97908	3.83	119192	79.95	117540	2.11	119398	81.02	119848	5.13	119870	5.748
800	159914	159914	116348	6.02	158828	189.36	157072	3.91	159344	190.84	159814	8.65	159914	11.744
1000	199946	199946	113568	8.53	198778	373.85	196974	6.58	199294	376.01	199940	11.64	199946	14.04
1200	239978	239978	101990	11	238848	641.86	236684	9.82	239386	644.91	239978	16.57	239978	23.954
1400	279984	279984	114798	14.82	278860	1025.55	276334	13.32	279276	1030.75	279976	25.38	279984	32.964
1600	319996	319996	95998	19.56	319106	1562.65	315948	18.09	319512	1626.32	319950	42.92	319996	50.524
1800	359992	359992	109598	24.25	358768	2188.28	355812	23.16	359404	2206.64	359992	44.88	359992	67.274
2000	400000	400000	147200	29.96	398864	3011.64	395484	28.65	399446	3031.73	399988	63.39	400000	67.148

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is not used in the Integrate heuristic,

Reduction procedure will be terminated if two consecutive loops are not improving

# A.11 Computation results for Data set 2 –R3

Size	Opt. Val. (HA)*	Best	HA**		MMX		SWAP/RFA		SWAP/MMX		INTEG		5-Runs***	
			value	time	value	time	value	time	value	time	value	time	value	time
18	4932	4932	4932	0.001	4548	0.001	4568	0.001	4932	0.01	4932	0.01	4932	0.01
50	14480	14480	14480	0.07	13900	0.05	13630	0.01	14026	0.06	14480	0.06	14480	0.07
100	29512	29512	29512	0.3	28856	0.38	28236	0.05	29220	0.39	29512	0.29	29512	0.298
200	59512	59512	58330	1.33	58616	2.75	57752	0.21	59130	2.83	59512	1.33	59512	1.334
400	119641	119638	109480	2.86	118456	23.93	116894	0.92	118880	24.25	119638	3.15	119638	4.688
600	179639	179634	165868	6.06	178028	79.93	176030	2.07	178856	80.75	179566	7.77	179634	10.088
800	239744	239744	208584	9.58	238180	189.43	235794	4.21	238798	190.84	239744	9.73	239744	11.958
1000	299786	299786	221250	10.75	298324	373.91	295236	6.65	298884	375.92	299674	14.98	299786	14.24
1200	359844	359844	264490	14.21	358170	641.73	354762	9.56	358856	644.57	359810	19.55	359844	22.588
1400	419898	419898	266346	17.64	418108	1025.88	414532	13.64	418836	1030.48	419884	26.95	419898	29.348
1600	479936	479936	259170	22.02	478400	1564.34	473910	17.86	479102	1584.5	479840	38.74	479936	40.25
1800	539962	539962	230980	25.76	538156	2185.99	533448	22.7	538984	2194.55	539962	38.32	539960	57.78
2000	599964	599956	242390	31.91	597862	3006.6	593008	30.42	599016	3019.66	599956	55.04	599948	92.434

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is not used in the Integrate heuristic,

Reduction procedure will be terminated if two consecutive loops are not improving

## A.12 Computation results for Data set 2 –R4

Size	Opt. Val. (HA)*	Best	HA**		MMX		SWAP/RFA		SWAP/MMX		INTEG		5-Runs***	
			value	time	value	time	value	time	value	time	value	time	value	time
18	6514	6514	6514	0.01	6514	0.01	6414	0.001	6514	0.01	6514	0.01	6514	0.01
50	19161	19158	13076	0.13	18774	0.05	18406	0.01	18938	0.06	19158	0.2	19158	0.188
100	39277	39208	32282	0.41	38416	0.37	37814	0.04	38826	0.39	39058	0.52	39208	0.568
200	79375	79358	73836	1.03	77976	2.75	77072	0.22	78562	2.82	79318	1.16	79358	1.268
400	159411	159334	153050	2.89	157922	23.9	155910	0.95	158684	24.18	159334	3.13	159334	3.706
600	239507	239504	213980	6.75	237654	80.09	234900	2.18	238558	80.98	239470	8.27	239504	9.132
800	319597	319560	292428	9.93	317432	189.43	314098	3.96	318512	191.62	319540	11.84	319560	17.028
1000	399606	399606	367638	15.09	398064	373.77	393062	6.57	398606	375.94	399606	15.16	399606	18.676
1200	479678	479678	391768	19.24	477650	641.34	472552	10.08	478492	644.69	479678	20.18	479678	26.562
1400	559709	559668	461374	24.73	557190	1025.52	552346	13.53	558578	1030.47	559636	31.39	559668	46.464
1600	639808	639808	506258	25.47	636948	1599.34	631726	18.57	638328	1566.36	639808	26.63	639808	36.104
1800	719856	719856	474306	29.54	717020	2185.61	711356	23.59	718428	2196.31	719856	33.74	719856	56.366
2000	799882	799882	439134	34.37	797340	3006.55	790596	29.4	798506	3012.79	799784	54.54	799882	78.506

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is not used in the Integrate heuristic,

Reduction procedure will be terminated if two consecutive loops are not improving

### A.13 Computation results for Data set 2 -R5

Size	Opt. Val.	Best	HA**		MMX		SWAP/RFA		SWAP/MMX		INTEG		5-Runs***	
	(HA)*		value	time	value	time	value	time	value	time	value	time	value	time
18	8142	8142	8142	0.01	8142	0.01	8020	0.01	8142	0.01	8142	0.01	8142	0.1
50	23956	23956	23956	0.14	23470	0.06	22710	0.01	23678	0.06	23956	0.14	23952	0.206
100	49046	49034	44222	0.54	47736	0.37	47444	0.05	48430	0.39	49034	0.59	49034	0.602
200	99137	99120	92222	1.73	97496	2.75	95884	0.19	97946	2.81	99120	1.82	99120	1.942
400	199220	199200	173330	4.41	197084	23.92	195172	0.9	198192	24.2	199194	4.71	199200	5.492
600	299364	299364	277446	7.24	297832	79.98	293606	2.2	298298	80.79	299364	7.26	299364	7.746
800	399352	399352	355432	12.37	396802	189.38	391796	4.1	397944	191.09	399226	14.77	399352	18.356
1000	499380	499362	452460	17.3	496114	373.82	491924	6.81	497874	376.07	499256	20.47	499362	26.762
1200	599492	599478	557530	21.69	596630	644	590604	9.77	598064	645.42	599290	26.07	599478	37.668
1400	699542	699484	605630	28.86	696932	1025.68	689894	13.46	698004	1030.82	699296	36.08	699484	49.786
1600	799600	799600	628724	29.23	795782	1586.83	789552	18.16	798070	1573.72	799600	31.12	799600	55.318
1800	899728	899696	681834	34.71	896304	2188.13	888834	23.34	898028	2194.88	899658	47.72	899696	72.016
2000	999728	999728	708810	38.84	996524	3018.64	988234	29.55	998138	3017.97	999622	55.38	999728	60.832

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is not used in the Integrate heuristic,

Reduction procedure will be terminated if two consecutive loops are not improving

#### A.14 Computation Results for Data set 3

Prob. No.	Opt. Val. *	Best Val.	HA**		MMX		SWAP/RFA		SWAP/USG		SWAP/MMX		INTEG		10-Runs***	
			value	time	value	time	value	time	value	time	value	time	value	time	value	time
A	9070	8990	7146	90.95	8964	125.42	8760	4.19	8738	4.54	8916	127.82	8982	176.79	8990	138.97
B	8514	8446	6368	60.42	8220	90.46	8254	3.32	8320	3.45	8336	92.45	8446	103.66	8446	108.17
C	6585	6516	5586	36.36	6402	59.89	6390	2.52	6342	2.66	6448	61.42	6516	49.1	6516	51.96
D	13688	13544	9954	144.21	13214	187.68	13254	5.56	13132	6.18	13406	190.88	13522	236.6	13544	252.24
E	12978	12848	10914	143.69	12466	169.77	12486	5.29	12530	5.54	12604	172.8	12838	192.56	12848	199.75
F	11582	11434	9338	93.51	11090	134.2	11236	4.48	11294	5.01	11268	136.81	11420	150.26	11434	158.28
G	15590	15438	11910	146.77	14944	210.54	15104	5.8	15164	7.39	15226	213.23	15434	221.26	15438	238.98

\* HA's optimal solution value

\*\* HA's symmetric subset value,

\*\*\* with randomly rearranged matrix

CPLEX is used in the Integrate heuristic if the problem size can be reduced to under 18x18

Reduction procedure will be terminated if two consecutive loops are not improving

**APPENDIX B**



## B.1 Component Grouping Procedures from [72]

- Step 1. Develop a component-to-component similarity matrix  $P$
- Step 2. Pick the largest element  $p_{ij}$  from  $P$ , and designate it as the present value of relationship counter  $RC$ .
- Step 3. Define a parameter  $U$ , where  $0 < U < 1$ , which is a measure of effectiveness of joining a component to a group consisting of other components. This parameter states the closeness an entering component must have with all the existing components within a group in order for the entering tool to join that group. The number of groups may differ based on the different values chosen for  $U$ .
- Step 4. Starting with the first row, examine each row for an elemental value that equals  $RC$ .
- Step 5. If none of the associated components in the row and column are already in a group, then form a group consisting of these two components and go to Step 7. If both components are already assigned to the same group, ignore and go to Step 7. If one of the components in the pair is in a group and the other one has not been assigned yet, go to Step 6(a). If both components are assigned to different groups, go to Step 6(b).
- Step 6. (a) Calculate the closeness ratio  $CR$  of the entering component with each group that has already been formed.  $CR$  is defined as the ratio of the total of all relationships the entering component has with the components that are currently in the group to the total number of components that are presently assigned to that group. The entering component is placed in a group that has the maximum closeness ratio  $MCR$ , as long as this maximum is greater than or equal to minimum threshold value  $MTV$ , calculated as  $U$  multiplied by the present value of  $RC$ . If the value of  $MCR$  is less than  $MTV$ , then a new group is formed consisting of two components having the relationship value that equals to the present value of  $RC$ . Go to Step 7.
- Step 6. (b) Duplication of one or more components is suggested. There are two possible alternatives that are checked in order of importance. The first alternative is to duplicate one additional component of either type (for illustrative purposes, designate the components in the pair as component A and B) and place it in the appropriate cell. The second alternative is to duplicate both components, one of each type, and either form a new group or place each in the appropriate existing groups. Four rules are suggested

- 1) Calculate the effect of duplicating one component. Check component A as the entering component for the groups where component B exists and B as the entering component for the groups where A exists. Determine the maximum closeness ratio,  $MCR$ , from all the groups that are checked and note the associated and note the associated group and entering component.
- 2) If  $MCR > RC \times U$ , the noted component is duplicated and assigned to the associated group. Go to Step 7.
- 3) If the maximum closeness ratio in the previous calculation was less than  $RC \times U$ , a check must be made to see if both components should be duplicated. From the previous calculations determine the  $MCR$  for the groups where A is the entering component ( $MCRA$ ) and the  $MCR$  for the groups where B is the entering component ( $MCRB$ ). Calculate the index value as maximum  $RC \times U/2$ . If both  $MCRA$  and  $MCRB$  are greater than the index value and  $|MCRA - MCRB| < RC \times U/2$ , duplicate both components and place each in an appropriate group. If either  $MCRA$  or  $MCRB$  is greater than the index value, regardless of the value of  $|MCRA - MCRB|$ , form a new group consisting of components A and B. Go to Step 7.
- 4) If none of the above conditions exists, ignore this observation and go to Step 7 since the contribution of any duplicating component in improving the efficiency of grouping is very limited.

Step 7. Check to see if all components are assigned to groups. If they are, go to Step 9, otherwise continue. Check the number of components in all groups. If the number of components in any group is equal to the number of slots then fathom the associated group. Fathoming a group means not allowing the group to be part of any further consideration that would add a new component to the group. Continue the check of the similarity matrix with the present value of  $RC$  proceeding sequentially in rows. If an element is found that is equal to the present value of  $RC$ , go to Step 5. If no such element is found, go to Step 8.

Step 8. Reduce the value  $RC$  to the next value in descending order of magnitude and return to Step 5.

Step 9. Assign each PCB to an appropriate group. This is accomplished by examining each PCB and assigning it to a group that has the most components it needs.



### B.3 Component-to-component similarity matrix from Appendix B.2

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

### B.4 Optimal Double feeder assignment for Appendix B.3

<i>i</i>	1	2	3	6	7	10	11	14	17	19	
<i>j</i>	5	9	4	12	8	15	13	16	18	20	total
<i>p<sub>ij</sub></i>	7	5	4	6	7	4	5	6	5	2	51

## B.5 Component and PCB grouping

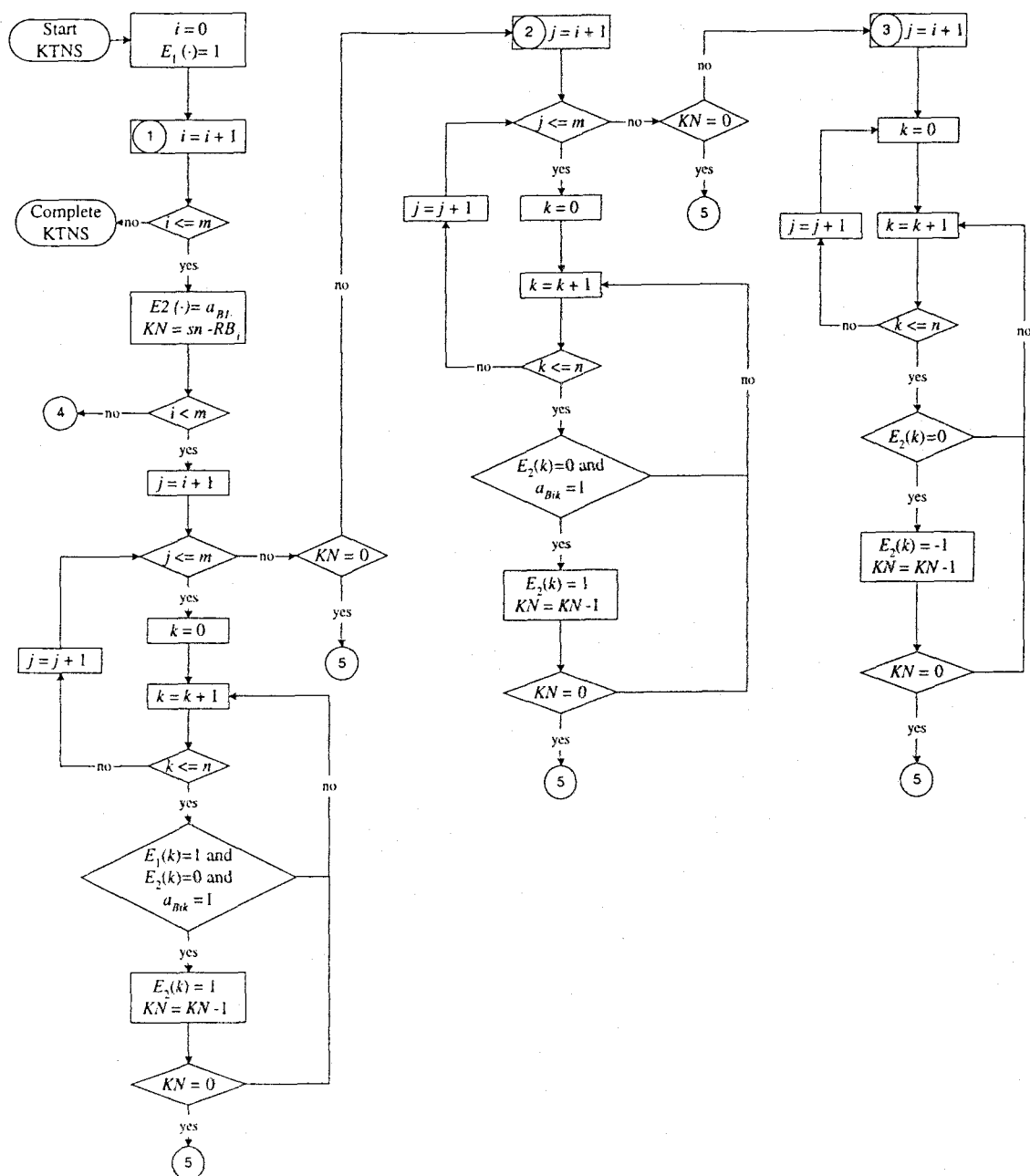
Set #	U	Component Group	PCB Group
1	0.00	7) 1 5 2 8 9 7 11 8) 7 8 9 10 11 14 16 9) 6 12 4 7 9 18 13 10) 17 18 3 4 5 8 6 11) 5 9 10 14 18 16 20 12) 19 14	7) 1 4 9 11 12 14 20 25 28 34 38 45 8) 3 6 15 19 27 30 36 40 9) 5 16 21 32 33 37 39 43 10) 2 17 22 24 31 11) 7 8 10 13 18 23 26 29 35 41 42 12) 44
2	0.25	1) 1 5 2 8 9 7 11 2) 7 8 9 10 11 14 16 3) 6 12 4 7 9 18 13 4) 17 18 15 10 14 16 20 5) 3 1 4 5 8 6 12 6) 19 14	1) 4 12 14 20 25 28 34 38 45 2) 3 6 15 19 23 27 36 40 3) 5 21 33 35 37 39 4) 8 10 13 18 24 26 29 30 41 42 5) 1 2 7 9 11 16 17 22 31 32 43 6) 44
3	0.50	1) 1 5 2 8 9 7 11 2) 7 8 9 10 11 14 16 3) 6 12 4 7 9 3 1 4) 18 9 17 15 10 14 16 5) 13 11 6 14 12 4 5 6) 20 17 1 5 18 11 12	1) 1 4 14 20 25 28 34 38 45 2) 3 6 15 19 36 40 3) 5 12 16 22 31 32 4) 8 10 13 23 26 33 35 42 5) 2 7 9 17 27 37 39 43 44 6) 11 18 21 24 29 30 41
4	0.75	1) 1 5 2 8 9 7 11 2) 7 8 9 10 6 12 14 3) 6 12 4 7 8 5 11 4) 11 8 14 5 6 12 4 5) 16 14 10 15 11 5 6 6) 18 9 17 16 15 5 12 7) 13 11 12 1 17 4 5 8) 3 1 4 5 10 2 7 9) 20 17 19	1) 3 4 20 25 38 45 2) 5 15 19 3) 16 32 4) 17 30 40 43 5) 6 7 13 23 27 36 6) 8 10 21 24 29 33 34 35 39 42 7) 9 11 28 37 8) 1 2 12 14 22 26 31 44 9) 18 41
5	1.00	1) 1 5 2 8 9 10 2) 7 8 1 5 11 9 3) 6 12 4 5 8 14 10 4) 9 8 2 6 11 7 5) 10 8 9 14 6 7 6) 11 8 14 5 7) 16 14 10 15 5 11 8) 4 5 6 7 8 9) 7 6 9 12 11 10) 18 9 15 10 6 11) 13 11 12 4 5 9 12) 17 18 5 9 11 12 13) 3 1 4 5 2 7 9 14) 16 9 11 12 15) 18 16 14 16) 20 17 17) 19 14	1) 45 2) 28 3) 7 15 17 43 4) 3 20 25 5) 5 19 6) 30 40 7) 6 23 36 8) 16 9) 27 32 10) 8 35 11) 9 37 39 12) 11 21 24 29 42 13) 1 2 4 12 14 22 26 31 38 14) 33 34 15) 10 13 16) 18 41 17) 44

Maximal Group Size (M.G.S.) allowed in component group: 7

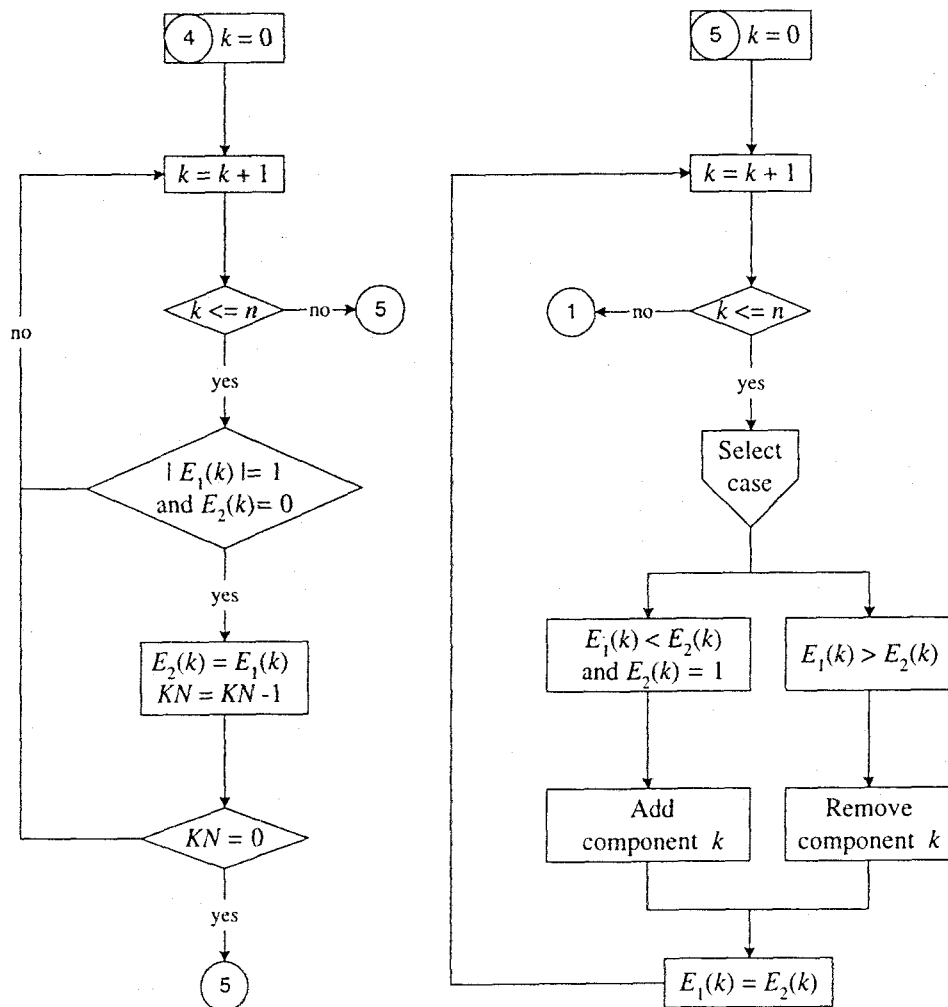
## B.6 Single feeder setup plan - IIGS

PCB seq.	slot number / component							cumulative setups	cumulative stops
	1	2	3	4	5	6	7		
24	17	18	3	4	5	6	14	1	1
2	2	18	3	4	5	6	14	2	2
22	1	18	3	4	5	7	14	4	3
31	1	8	3	4	5	9	14	6	4
17	12	8	3	4	5	9	14	7	5
34	12	8	11	16	5	9	2	10	6
25	12	8	11	13	5	9	2	11	7
28	1	8	11	13	5	9	2	12	8
38	1	8	11	7	5	9	2	13	9
4	1	8	11	7	5	9	2	13	9
20	1	8	11	6	5	9	2	14	10
45	1	8	10	18	5	9	2	16	11
12	1	2	3	18	5	9	10	17	12
1	1	2	3	18	5	9	10	17	12
14	1	2	7	18	5	15	4	20	13
9	1	13	11	18	5	15	4	22	14
11	1	13	11	19	5	17	12	25	15
37	4	13	11	19	6	17	12	27	16
43	4	10	14	19	6	17	12	29	17
16	4	10	7	8	6	17	12	31	18
5	4	10	7	8	6	9	12	32	19
39	15	10	7	13	6	9	12	34	20
33	17	16	7	13	6	9	12	36	21
21	17	18	7	16	6	9	12	37	22
32	17	18	7	11	6	3	12	39	23
27	17	14	7	11	6	3	12	40	24
40	17	14	7	11	8	3	12	41	25
30	17	14	7	11	8	20	12	42	26
3	9	14	7	11	8	20	12	43	27
19	9	14	7	10	8	6	12	45	28
15	9	14	7	10	8	16	12	46	29
36	5	14	7	10	8	16	12	47	30
6	16	14	15	10	8	11	12	49	31
7	5	14	15	10	8	6	12	51	32
8	5	14	15	10	3	9	18	54	33
35	5	14	15	10	6	9	18	55	34
42	5	14	15	16	17	9	18	57	35
23	11	14	15	16	17	9	18	58	36
13	11	14	15	16	17	9	18	58	36
29	11	14	15	16	17	20	18	59	37
18	19	14	15	16	17	20	18	60	38
10	19	14	15	16	17	20	18	60	38
41	19	1	15	16	17	20	18	61	39
26	4	1	15	19	17	20	10	63	40
44	4	13	3	19	17	20	14	66	41

## B.7 KTNS pseudo-code flowchart

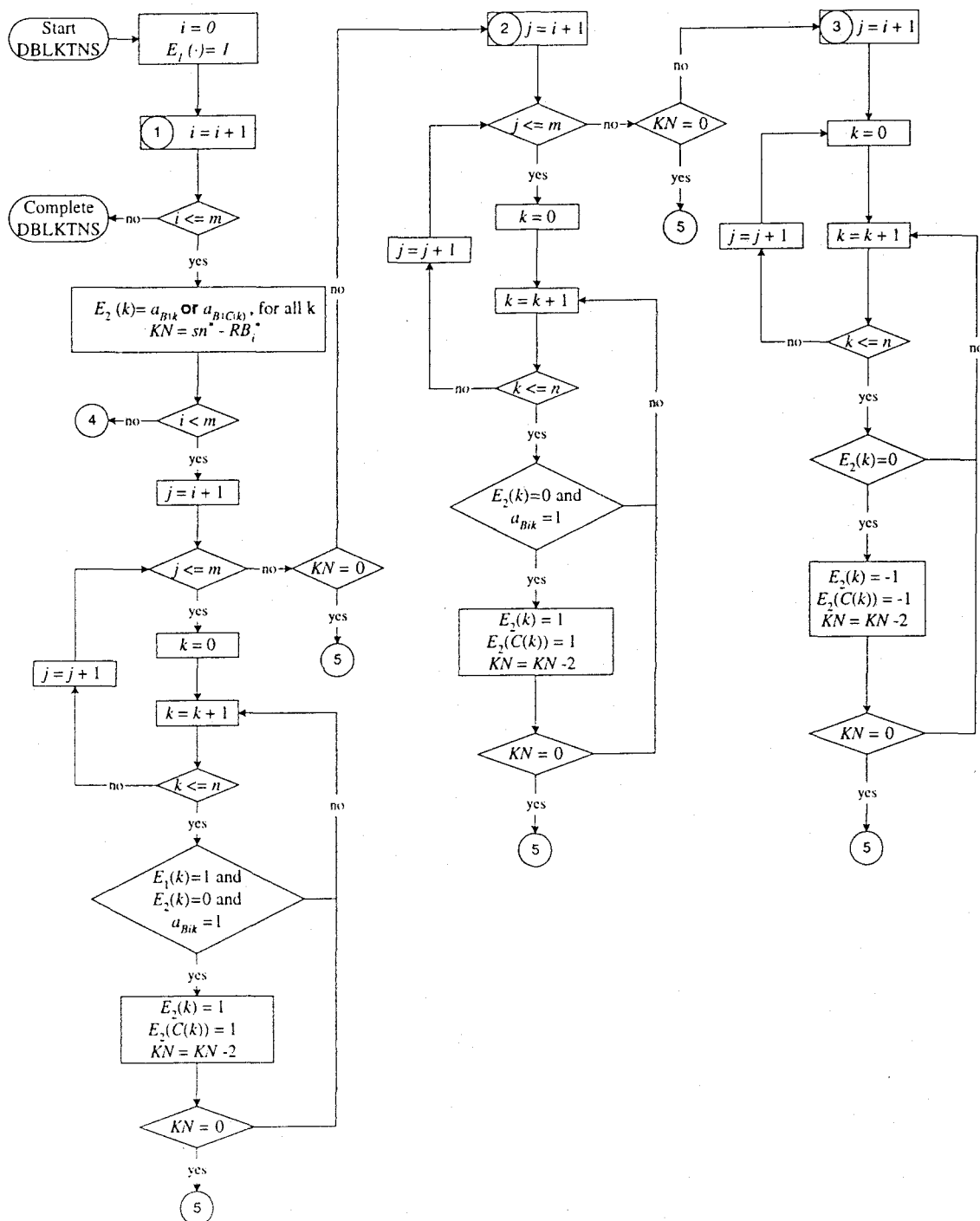


## B.7 KTNS pseudo-code flowchart (Continued)

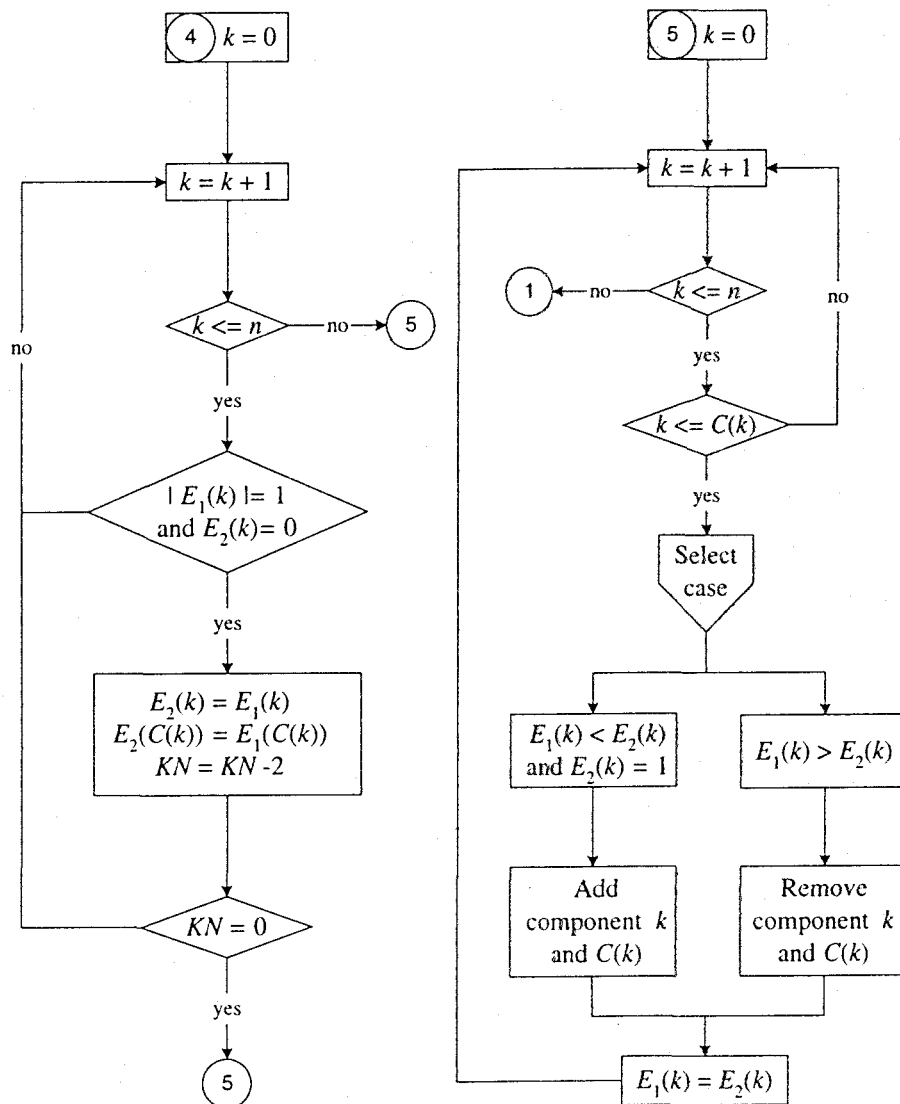




### B.8 Modified KTNS pseudo-code flowchart for double-feeder



# B.8 Modified KTNS pseudo-code flowchart for double-feeder (Continued)



### B.9 Single feeder setup plan – IIGS/KTNS

PCB seq.	slot number / component							cumulative setups	cumulative stops
	1	2	3	4	5	6	7		
24	2	3	5	6	14	17	18	0	0
2	2	3	5	6	14	4	18	1	1
22	2	3	5	1	14	4	7	3	2
31	8	3	5	1	14	4	9	5	3
17	8	12	5	1	14	4	9	6	4
34	8	12	5	2	11	16	9	9	5
25	8	12	5	2	11	13	9	10	6
28	8	1	5	2	11	13	9	11	7
38	8	1	5	2	11	7	9	12	8
4	8	1	5	2	11	7	9	12	8
20	8	1	5	2	11	6	9	13	9
45	8	1	5	2	10	18	9	15	10
12	8	1	5	2	3	9	10	16	11
1	8	1	5	2	3	9	10	16	11
14	4	1	5	2	7	10	15	19	12
9	4	1	5	13	7	10	11	21	13
11	17	1	5	13	19	12	11	24	14
37	17	6	4	13	19	12	11	26	15
43	14	6	4	13	19	12	10	28	16
16	7	6	4	8	13	12	10	30	17
5	7	6	4	8	9	12	10	31	18
39	7	6	13	8	9	12	15	33	19
33	7	6	13	17	9	12	16	35	20
21	7	6	18	17	9	12	16	36	21
32	7	6	11	17	9	12	3	38	22
27	7	6	11	17	9	14	3	39	23
40	7	8	11	17	9	14	3	40	24
30	7	8	11	17	9	14	20	41	25
3	7	8	11	17	9	14	20	41	25
19	7	8	11	6	9	14	10	43	26
15	7	8	11	16	12	14	10	45	27
36	7	8	11	10	5	14	16	46	28
6	15	8	11	10	5	14	16	47	29
7	5	8	6	10	12	14	15	49	30
8	5	18	6	9	10	3	15	52	31
35	5	18	6	9	10	3	15	52	31
42	5	18	16	9	10	17	15	54	32
23	11	18	16	9	14	17	15	56	33
13	11	18	16	20	14	17	15	57	34
29	11	18	16	20	14	17	15	57	34
18	19	18	16	20	14	17	15	58	35
10	19	18	16	20	14	17	15	58	35
41	19	1	4	20	14	17	15	60	36
26	19	1	4	10	14	17	15	61	37
44	19	1	4	10	14	3	13	63	38

### B.10 Double feeder setup plan - HGS/Modified KTNS and optimal assignment

PCB	slot number / components							cumulative	cumulative
seq.	1	2	3	4	5	6	7	setups	stops
24	1 5	2 9	3 4	6 12	7 8	14 16	17 18	0	0
2	1 5	2 9	3 4	6 12	7 8	14 16	17 18	0	0
22	1 5	2 9	3 4	6 12	7 8	14 16	17 18	0	0
31	1 5	2 9	3 4	6 12	7 8	14 16	17 18	0	0
17	1 5	2 9	3 4	6 12	7 8	14 16	17 18	0	0
34	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
25	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
28	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
38	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
4	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
20	1 5	2 9	11 13	6 12	7 8	14 16	17 18	1	1
45	1 5	2 9	11 13	6 12	7 8	10 15	17 18	2	2
12	1 5	2 9	11 13	6 12	7 8	10 15	3 4	3	3
1	1 5	2 9	11 13	6 12	7 8	10 15	3 4	3	3
14	1 5	2 9	11 13	6 12	7 8	10 15	3 4	3	3
9	1 5	2 9	11 13	6 12	7 8	10 15	3 4	3	3
11	1 5	17 18	11 13	6 12	19 20	10 15	3 4	5	4
37	1 5	17 18	11 13	6 12	19 20	10 15	3 4	5	4
43	14 16	17 18	11 13	6 12	19 20	10 15	3 4	6	5
16	14 16	17 18	11 13	6 12	7 8	10 15	3 4	7	6
5	14 16	2 9	11 13	6 12	7 8	10 15	3 4	8	7
39	14 16	2 9	11 13	6 12	7 8	10 15	3 4	8	7
33	14 16	2 9	11 13	6 12	7 8	17 18	3 4	9	8
21	14 16	2 9	11 13	6 12	7 8	17 18	3 4	9	8
32	14 16	2 9	11 13	6 12	7 8	17 18	3 4	9	8
27	14 16	2 9	11 13	6 12	7 8	17 18	3 4	9	8
40	14 16	2 9	11 13	6 12	7 8	17 18	3 4	9	8
30	14 16	2 9	11 13	6 12	7 8	17 18	19 20	10	9
3	14 16	2 9	11 13	6 12	7 8	17 18	19 20	10	9
19	14 16	2 9	11 13	6 12	7 8	17 18	10 15	11	10
15	14 16	2 9	11 13	6 12	7 8	17 18	10 15	11	10
36	14 16	2 9	11 13	6 12	7 8	1 5	10 15	12	11
6	14 16	2 9	11 13	6 12	7 8	1 5	10 15	12	11
7	14 16	2 9	11 13	6 12	7 8	1 5	10 15	12	11
8	14 16	2 9	3 4	6 12	17 18	1 5	10 15	14	12
35	14 16	2 9	3 4	6 12	17 18	1 5	10 15	14	12
42	14 16	2 9	3 4	11 13	17 18	1 5	10 15	15	13
23	14 16	2 9	3 4	11 13	17 18	1 5	10 15	15	13
13	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
29	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
18	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
10	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
41	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
26	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14
44	14 16	19 20	3 4	11 13	17 18	1 5	10 15	16	14

### B.11 Evaluation results by different grouping threshold values and methods on example data

method	setup				stop				CPU time			
	max	min	mean	stdev	max	min	mean	stdev	max	min	mean	stdev
M2	126	108	114.6	5.6	44	40	42.1	1.4	0.02	0.02	0.02	0.00
M5	34	26	30.1	2.6	29	21	24.8	2.4	0.03	0.02	0.02	0.00
M7	44	34	38.2	3.2	36	26	29.9	2.9	0.06	0.02	0.02	0.01

The average similarity value of random double assignment: 54 with standard deviation 5.4

method	setups				
	<i>U</i>				
	0.00	0.25	0.50	0.75	1.00
M1	66	70	67	67	77
M3	63	65	64	62	68
M4	16	18	18	18	17
M6 (avg)	21.4	23.0	22.7	21.9	23.7
M6 (dev)	2.1	1.5	2.5	2.1	2.1

method	stops				
	0.00	0.25	0.50	0.75	1.00
M1	41	41	40	41	41
M3	38	39	40	39	39
M4	14	17	18	16	14
M6 (avg)	18.4	19.1	19.7	19.0	20.6
M6 (dev)	1.3	2.1	2.1	2.2	1.8

method	CPU time (sec)				
	0.00	0.25	0.50	0.75	1.00
M1	1.40	1.38	1.32	1.24	1.20
M3	0.03	0.02	0.02	0.03	0.02
M4	0.03	0.03	0.03	0.03	0.03
M6 (avg)	0.03	0.03	0.03	0.03	0.02
M6 (dev)	0.00	0.00	0.01	0.01	0.01

method:

- M1. IIGS sequence without KTNS policy (IIGS)
- M2. Random Board Sequence with KTNS policy (RBS/KTNS)
- M3. IIGS sequence with KTNS policy ( IIGS/KTNS)
- M4. IIGS sequence with modified KTNS and Optimal Feeder Assignment (IIGS/OFA DBLKTNS)
- M5. Random PCB sequence with modified KTNS and optimal assignment (RBS/OFA DBLKTNS)
- M6. IIGS sequence with modified KTNS but Random Feeder Assignment (IIGS/RFA DBLKTNS)
- M7. Random sequence and assignment with modified KTNS (RBS/RFA DBLKTNS)

### B.12 Summary results and effects evaluation of industry data sets

method	Data set													
	A		B		C		D		E		F		G	
	setup	stop	setup	stop	setup	stop	setup	stop	setup	stop	setup	stop	setup	stop
M1	2034	225	1916	247	1276	152	3213	398	2898	333	2667	326	3725	460
M2	2841	342	2292	342	1524	211	4625	598	4081	501	3364	477	5266	682
M3	1178	172	952	196	680	106	1679	344	1569	246	1308	258	2014	373
M4	405	120	272	123	177	70	676	256	548	185	456	190	789	282
M5	693	230	522	232	293	128	1357	461	1121	362	891	347	1559	512
M6	769	162	538	162	332	82	1243	309	1075	242	855	223	1587	351
M7	1760	323	1271	300	729	175	3246	573	2788	478	2139	435	3889	654
RFA/OFA	11.9%		12.0%		12.3%		9.4%		10.2%		10.0%		9.5%	
size	437×696		458×626		281×548		744×796		620×770		608×712		843×826	
M4 CPUtime	262		282		116		701		498		493		888	

\* results are presented in percentage

data set		Effect								
		E1	E2	E3	E4	E5	E6	E7	E8	E9
		M1-M3	M2-M3	M7-M6	M7-M5	M2-M7	M2-M5	M3-M4	M7-M4	M2-M4
A	setup	42.1	58.5	56.3	60.6	38.1	75.6	65.6	77.0	85.7
	stop	23.6	49.6	49.8	28.8	5.6	32.8	30.2	62.8	64.9
B	setup	50.3	58.5	57.7	58.9	44.5	77.2	71.4	78.6	88.1
	stop	20.6	42.7	45.9	22.5	12.3	32.1	37.2	59.0	64.0
C	setup	46.7	55.4	54.5	59.9	52.1	80.8	74.0	75.7	88.4
	stop	30.3	49.8	53.0	26.8	17.3	39.5	34.0	59.9	66.8
D	setup	47.7	63.7	61.7	58.2	29.8	70.7	59.7	79.2	85.4
	stop	13.6	42.5	46.1	19.5	4.3	22.9	25.6	55.3	57.2
E	setup	45.9	61.5	61.5	59.8	31.7	72.5	65.1	80.3	86.6
	stop	26.1	50.9	49.5	24.3	4.6	27.8	24.8	61.3	63.1
F	setup	51.0	61.1	60.0	58.4	36.4	73.5	65.1	78.7	86.4
	stop	20.9	45.9	48.9	20.4	8.7	27.3	26.4	56.4	60.2
G	setup	45.9	61.8	59.2	59.9	26.2	70.4	60.8	79.7	85.0
	stop	18.9	45.3	46.3	21.8	4.1	25.0	24.4	56.9	58.7
avg	setup	47.1	60.1	58.7	59.4	37.0	74.4	66.0	78.5	86.5
	stop	22.0	46.7	48.5	23.4	8.1	29.6	28.9	58.8	62.1
dev	setup	3.0	2.8	2.7	0.9	9.0	3.8	5.2	1.6	1.3
	stop	5.3	3.5	2.6	3.4	5.0	5.6	5.0	2.8	3.5

**APPENDIX C**

# C.1 2<sup>4</sup> Factorial Experiment Design with one replicate and Results

run	block	USG stdev	REQ stdev.	Sequence	Assignment	Feeder setup	CPU time
1	1	low	low	RBS	RFA	2164	106
2	1	high	low	RBS	RFA	385	140
3	1	low	high	RBS	RFA	3131	107
4	1	high	high	RBS	RFA	254	164
5	1	low	low	IIGS	RFA	166	179
6	1	high	low	IIGS	RFA	146	180
7	1	low	high	IIGS	RFA	245	206
8	1	high	high	IIGS	RFA	132	173
9	1	low	low	RBS	OFA	1048	32
10	1	high	low	RBS	OFA	124	105
11	1	low	high	RBS	OFA	1237	34
12	1	high	high	RBS	OFA	131	190
13	1	low	low	IIGS	OFA	124	153
14	1	high	low	IIGS	OFA	124	129
15	1	low	high	IIGS	OFA	125	199
16	1	high	high	IIGS	OFA	124	150
17	2	low	low	RBS	RFA	2097	107
18	2	high	low	RBS	RFA	254	160
19	2	low	high	RBS	RFA	3194	106
20	2	high	high	RBS	RFA	220	170
21	2	low	low	IIGS	RFA	151	180
22	2	high	low	IIGS	RFA	124	190
23	2	low	high	IIGS	RFA	166	201
24	2	high	high	IIGS	RFA	124	169
25	2	low	low	RBS	OFA	1025	32
26	2	high	low	RBS	OFA	125	154
27	2	low	high	RBS	OFA	1269	31
28	2	high	high	RBS	OFA	139	180
29	2	low	low	IIGS	OFA	124	160
30	2	high	low	IIGS	OFA	124	139
31	2	low	high	IIGS	OFA	124	210
32	2	high	high	IIGS	OFA	124	169



## C.2 Analysis of Variance for log(Feeder setup), Full model

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:USG stdev.	11.2378	1	11.2378	1133.16	0.0000*
B:REQ stdev.	0.0422074	1	0.0422074	4.26	0.0557
C:Sequence	16.0456	1	16.0456	1617.95	0.0000*
D:Assignment	1.96767	1	1.96767	198.41	0.0000*
AB	0.145916	1	0.145916	14.71	0.0015*
AC	8.48334	1	8.48334	855.41	0.0000*
AD	0.0757417	1	0.0757417	7.64	0.0138
BC	0.00450673	1	0.00450673	0.45	0.5099
BD	0.000133122	1	0.000133122	0.01	0.9092
CD	0.65456	1	0.65456	66.00	0.0000*
ABC	0.0297209	1	0.0297209	3.00	0.1027
ABD	0.0911924	1	0.0911924	9.20	0.0079*
ACD	0.0252105	1	0.0252105	2.54	0.1304
BCD	0.0146639	1	0.0146639	1.48	0.2416
blocks	0.0463862	1	0.0463862	4.68	0.0461
Total error	0.158676	16	0.00991727		
Total (corr.)	39.0234	31			

\*:significant at 99% confidence level

R-squared = 99.6%

R-squared (adjusted for d.f.) = 99.2%

## C.3 Backward selection stepwise regression for log(Feeder setup).

Step	Removed variable	F-value	number of variables		R <sup>2</sup>	adj. R <sup>2</sup>	MSE
			in the model	df			
0			15	16	99.50%	99.03%	0.01222
1	BD	0.01090	14	17	99.50%	99.09%	0.01151
2	DC	0.39173	13	18	99.49%	99.12%	0.01112
3	ABCD	0.86520	12	19	99.46%	99.12%	0.01104
4	BCD	1.32862	11	20	99.43%	99.11%	0.01122
5	ACD	2.24726	10	21	99.36%	99.06%	0.01189
6	ABC	2.50079	9	22	99.28%	98.99%	0.01270
7	B	3.32463	8	23	99.18%	98.89%	0.01398
8	AD	5.41845	7	24	98.98%	99.69%	0.01655
9	ABD	5.50946	6	25	98.75%	99.45%	0.01954
10	AB	7.46847	5	26	98.37%	98.06%	0.02440

#### C.4 Analysis of Variance for CPU times, Full model

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:USG stdev.	8417.53	1	8417.53	87.53	0.0000*
B:REQ stdev.	3061.53	1	3061.53	31.83	0.0000*
C:Sequence	29342.5	1	29342.5	305.11	0.0000*
D:Assignment	6932.53	1	6932.53	72.08	0.0000*
AB	16.5313	1	16.5313	0.17	0.6839
AC	25144.0	1	25144.0	261.45	0.0000*
AD	1391.28	1	1391.28	14.47	0.0016*
BC	13.7813	1	13.7813	0.14	0.7100
BD	1313.28	1	1313.28	13.66	0.0020*
CD	552.781	1	552.781	5.75	0.0291
ABC	2194.53	1	2194.53	22.82	0.0002*
ABD	357.781	1	357.781	3.72	0.0717
ACD	4347.78	1	4347.78	45.21	0.0000*
BCD	75.0313	1	75.0313	0.78	0.3902
blocks	385.031	1	385.031	4.00	0.0627
Total error	1538.75	16	96.1719		
Total (corr.)	85084.7	31			

R-squared = 98.2%      R-squared (adjusted for d.f.) = 96.7%

#### C.5 Analysis of Variance for CPU times, Reduced model

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
A:USG stdev.	8417.53	1	8417.53	69.19	0.0000*
B:REQ stdev.	3061.53	1	3061.53	25.17	0.0001*
C:Sequence	29342.5	1	29342.5	241.20	0.0000*
D:Assignment	6932.53	1	6932.53	56.99	0.0000*
AC	25144.0	1	25144.0	206.69	0.0000*
AD	1391.28	1	1391.28	11.44	0.0028*
BD	1313.28	1	1313.28	10.80	0.0035*
ABC	2194.53	1	2194.53	18.04	0.0004*
ACD	4347.78	1	4347.78	35.74	0.0000*
blocks	385.031	1	385.031	3.17	0.0897
Total error	2554.66	21	121.65		
Total (corr.)	85084.7	31			

\*:significant at 99% confidence level  
R-squared = 97.0%      R-squared (adjusted for d.f.) = 95.8%

### C.6 Backward selection stepwise regression for CPU times.

Step	Removed variable	F-value	number of variables in the model	df	R <sup>2</sup>	adj. R <sup>2</sup>	MSE
0			15	16	97.81%	95.76%	116.281
1	BC	0.11852	14	17	97.80%	95.98%	110.252
2	AB	0.14994	13	18	97.78%	96.17%	105.045
3	ABCD	0.60242	12	19	97.70%	96.25%	102.844
4	BCD	0.72954	11	20	97.62%	96.30%	101.456
5	ABD	3.52645	10	21	97.19%	95.86%	113.662
6	CD	4.86337	9	22	96.54%	95.13%	133.622