T.C.
MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

OPTIMIZING THE OPERATIONS OF ELECTRONIC COMPONENT PLACEMENT MACHINES

Ali Fuat ALKAYA

THESIS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ENGINEERING MANAGEMENT

SUPERVISOR
Prof. Dr. M. Akif EYLER

CO-SUPERVISOR
Assoc. Prof. Dr. Ekrem DUMAN

ISTANBUL 2009

T.C.
MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

OPTIMIZING THE OPERATIONS OF ELECTRONIC COMPONENT PLACEMENT MACHINES

Ali Fuat ALKAYA
(142202420020299)

THESIS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ENGINEERING MANAGEMENT

SUPERVISOR
Prof. Dr. M. Akif EYLER

CO-SUPERVISOR
Assoc. Prof. Dr. Ekrem DUMAN

ISTANBUL 2009

MARMARA UNIVERSITY

THE INSTITUTE FOR

GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

ACCEPTANCE AND APPROVAL DOCUMENT

The jury established by the Executive Board of the *INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES* on 27.04.2009 (Resolution no:2009/11-03) has accepted Mr. Ali Fuat ALKAYA's thesis titled "OPTIMIZING THE OPERATIONS OF ELECTRONIC COMPONENT PLACEMENT MACHINES" as Doctor of Philosophy thesis in Engineering Management.

Advisor	:Prof. Dr. M. Akif EYLER
1. Member of the jury	:Prof. Dr. Fuat İNCE
2. Member of the jury	:Asst. Prof. Dr. Serol BULKAN
3. Member of the jury	:Asst. Prof. Dr. Şule ÖNSEL
4. Member of the jury	:Asst. Prof. Dr. Tunç BOZBURA

Date : 08.06.2009

APPROVAL

Ms. Ali Fuat ALKAYA has satisfactorily completed the requirements for the degree of Doctor of Philosophy in Engineering Management at Marmara University. The Executive Committee approves that he be granted the degree of Doctor of Philosophy on .... (Resolution no:………).

DIRECTOR OF THE INSTITUTE

Prof. Dr. Sevil ÜNAL

PREFACE

To my father, Ali ALKAYA.

June, 2009 **Ali Fuat ALKAYA**

ACKNOWLEDGMENT

I would like to thank my academic supervisor, Dr. Ekrem DUMAN, for his encouragement, kindness, support, patience and valuable guidance throughout this study. I am also grateful to Dr. M. Akif EYLER for his acceptance my thesis for co-supervising. I will benefit from his suggestions in my whole life**.**

I wish to express my heartfelt thanks to my wife, Reyhan ALKAYA, for her endless support and patience throughout my graduate education.

To my mother, my sister and my brothers, many thanks for your continuous support and understanding.

Many thanks to my colleagues in the Computer Science and Engineering Department for their sincere friendship and making my time spent at the Marmara University an enjoyable experience.

I also gratefully acknowledge the support of Turkish Scientific and Technological Research Institution (TÜBITAK) in this research, through the research project 108M198**.**

June, 2009 **Ali Fuat ALKAYA**

CONTENTS

ABSTRACT

Optimizing the Operations of Electronic Component Placement Machines

The extensive usage of printed circuit boards (PCBs) in numerous electronic products has placed an unparalleled demand for PCBs. Among several operations in a surface mount technology assembly line, the operations of placement machines are the bottleneck of the line. Main optimization problems of placement machines are determining the placement sequence of components (placement sequencing problem) and assignment of component types to feeder cells (feeder configuration problem). These problems turn out to be combinatorial optimization problems that are NP-Complete thereby, in general, restricting optimal solution techniques to small instances.

This thesis focuses on optimizing the operations of specific placement machines by determining near optimal placement sequences and feeder configurations. Specifically, these machines are the chip mounter and the chip shooter placement machines where both of them have a rotational turret. The inherent design of the machines entails concurrent solution of these problems. This integrated problem is called as the assembly time minimization problem.

The research begins by an extended survey on chip mounter and chip shooter machines. The survey points out the varying turret time property of these machines which is mostly overlooked by the researchers. A new generalization of Traveling Salesman Problem (TSP) is introduced to the literature and called as the Sequence Dependent TSP (SDTSP). The thesis covers nonlinear integer programming formulation of the SDTSP. Next, it is shown that the placement sequencing problem of both machines turn out to be a SDTSP. Placement sequencing and feeder configuration problems for both machines are formulated. Furthermore, the combined assembly time minimization problems for both machines are formulated as nonlinear integer programming formulations.

This thesis continues by investigating the optimization opportunities of chip mounter machines. Since it is very time consuming to solve the mathematical models for optimal solutions, several heuristics are developed. These heuristics are iATMA, Adjust First Point Procedure (AFPP), Postpone Deviant (PD), Group

Insertion (GI) and Individual Insertion (II). Record-to-Record Travel with Local Exchange Moves (RRTLEM) and Pair-wise Exchange Procedures (PEP) are local search methods that are also implemented. We randomly generated PCB data with various number of components. Computational results are presented to demonstrate the effectiveness of these heuristics.

June, 2009 **Ali Fuat ALKAYA**

ÖZET

Elektronik Parça Yerleştirme Makinelerinin Optimizasyonu

Basılı devre kartlarının günümüzde çok sayıda elektronik üründe kullanılması onlara olan talebi hiç görülmemiş bir şekilde artırmıştır. Yüzey yapıştırma teknolojisini kullanan montaj hatlarındaki darboğaz olan kaynak genellikle dizgi makineleridir. Dizgi makinelerindeki ana eniyileme problemleri ise parça montaj sırasının belirlenmesi (montaj sıralaması) ve parça tiplerinin besleme hücrelerine bölüştürülmesidir (besleme konfigürasyonu). Bu problemler NP-Zor olan birleşimsel eniyileme problemlerine dönüşür ve bu da genelde eniyiyi bulan çözüm tekniklerini sadece küçük boyuttaki örneklerle sınırlandırır.

Bu tez belirli dizgi makinelerinin işlemlerini eniyilemeye odaklanmıştır ve bunu eniyiye yakın montaj sıralaması ve besleme konfigürasyonlarını bularak yapacaktır. Bu makineler "çip parça yerleştirici" ve "çip saçıcı" ismiyle anılan makinelerdir ve her ikisi de döner tarete sahiptir. Makinelerin dizaynı bu iki problemin beraber çözülmesini gerektirmektedir. Bu bütünleşmiş probleme dizgi zamanının en aza indirilmesi problemi denmektedir.

Araştırma çip parça yerleştirici ve çip saçıcı makineler üzerinde kapsamlı bir yazın incelemesi yaparak başlamaktadır. İnceleme, bu makinelerin değişkenlik gösteren taret zamanı özelliklerinin çoğu araştırmacı tarafından göz ardı edildiğini ortaya koymaktadır. Bununla alakalı olarak Gezgin Satıcı Probleminin (GSP) yeni bir genellemesi yazına kazandırılmış ve Sıraya Dayalı GSP (SDGSP) olarak adlandırılmıştır. SDGSP için doğrusal olmayan tamsayılı programlama formulasyonu da verilmiştir. Sonrasında, her iki makine için de montaj sıralama problemlerinin SDGSP'ne dönüştüğü gösterilmiştir. Yine her iki makine için de besleme konfigürasyonu problemi formüle edilmiştir. Bundan başka, her iki makine için dizgi zamanının en aza indirilmesi problemi doğrusal olmayan tamsayılı programlamayla formüle edilmiştir.

Tez çip parça yerleştirici makinelerinin eniyileme çözümlerinin araştırılmasıyla devam etmektedir. En iyi çözümleri elde etmek için matematiksel modellerin çözülmesi çok zaman alacağından birçok sezgisel yöntem geliştirilmiştir. Geliştirilen bu sezgisel yöntemler, iATMA, İlk Noktayı Değiştirme Prosedürü

(İNDP), Ayrıkları Erteleme (AE), Grup Ekleme (GE) ve Bireysel Ekleme (BE)'dir. Ayrıca Kayıttan Kayıta Yerel Değiştirme Hareketleriyle Dolaşma (KKYDHD) ve Çift olarak Değiştirme Prosedürleri (ÇDP) uygulanan yerel arama metotlarıdır. Çeşitli parça sayısına sahip rassal BDK verileri oluşturulmuştur. Hesaplama sonuçları öne sürülen sezgisel yöntemlerin etkili olduğunu göstermektedir.

Haziran, 2009 **Ali Fuat ALKAYA**

CLAIM FOR ORIGINALITY

Optimizing the Operations of Electronic Component Placement Machines

Within this technology era, printed circuit boards (PCBs) are used in almost every technological device. They are produced by placement machines which are generally the bottleneck of a PCB assembly line. Since the performance of the placement machines studied in this thesis depends heavily on both the placement sequencing and feeder configuration problems, they were studied and solved. The significant original contributions of this thesis are concluded in the following:

A comprehensive survey on chip shooter placement machines was carried out. It is pointed out that varying turret rotation time is an important property of chip shooters but very few of the studies propose solution procedures considering this property. For both placement machines, including the chip mounter machine and the chip shooter machine, the placement sequencing problem which takes into account the varying turret rotation time have been formulated as nonlinear integer programming model. We pointed out that the problem arising due to varying rotational turret time values of a chip shooter is actually a new generalization of the TSP. To the best of our knowledge this TSP generalization has not been defined in the literature and we named it as the Sequence Dependent TSP (SDTSP). Regarding chip mounter placement machines, an original linear integer programming model has been formulated for the feeder configuration problem. Besides, the assembly time minimization problem has been formulated as an original nonlinear integer programming model. A secondary problem is formulated for chip mounter machines. Various heuristics are proposed and applied to optimize the assembly process of chip mounter machines. These are Inverse ATMA (iATMA), Adjust First Point Procedure (AFPP), Adjust First Point according to Total Cost Procedure (AFPTCP), Postpone Deviant (PD) and Extended PD (EPD).

June, 2009	**Supervisors**	**Student**
	Prof. Dr. M. Akif EYLER	**Ali Fuat ALKAYA**
	Assoc. Prof. Dr. Ekrem DUMAN	

SYMBOLS

$\boldsymbol{C}$	: capacity of a vehicle in vehicle routing problem
$\boldsymbol{C2_{rs}}$	: time required for the feeder carriage to travel from feeder *r* to feeder *s*
$\boldsymbol{C_{ijp}}$	: cost of travel from point *i* to point *j* when point *j* is visited in p^{th} position
$\boldsymbol{c_e}$	: cost of edge *e*
$\boldsymbol{c_{ij}}$	: cost of travel from city *i* to city *j*
$\boldsymbol{ct_{it}}$	: component type matrix ($N \ x \ n$) indicating the component type for each component
$\boldsymbol{D_{ijp}}$	: the dominating factor in C_{ijp} definition
$\boldsymbol{D}$	: distance matrix for distances between offices
$\boldsymbol{d_{kl}}$	: the distance between office *k* and office *l*
$\boldsymbol{d_i}$	: demand at each node in vehicle routing problem
$\boldsymbol{E}$	: set of edges
$\boldsymbol{e}$	: edge
$\boldsymbol{EP}$	: set of excluded points
$\boldsymbol{ft_j}$	: free time value for point *j*
$\boldsymbol{G}$	: graph consisting of edges and vertices
$\boldsymbol{g_{tk}}$	: group (weight category) matrix ($n \ x \ K$) indicating the group for each component type
$\boldsymbol{H}$	: number of heads
$\boldsymbol{K}$	: number of weight categories
$\boldsymbol{M}$	: number of vehicles in vehicle routing problem
$\boldsymbol{N}$	: total number of components to be placed,
$\boldsymbol{n}$	: number of component types
$\boldsymbol{N_k}$	: number of components in each weight category *k.*
$\boldsymbol{n_k}$	: number of component types in each weight category *k*.
noi	:number of iterations
npz	: number of heads in no pickup zone
$\boldsymbol{p}$	: is the placement order or placement position.
$\boldsymbol{R}$	: number of feeder slots

$\boldsymbol{tt_k}$	: turret rotation time for each weight category *k*.
$\boldsymbol{t^0}$	: time required to retrieve a component from the feeder magazine and concurrently populate a component on the PCB after all the mechanisms are positioned
$\boldsymbol{SC_i}$	: the number of rotational steps of the turret in a speed category *i*
$\boldsymbol{S}$	: set of feasible solutions for a combinatorial optimization problem
$\boldsymbol{s}$	: a solution for a combinatorial optimization problem
$\boldsymbol{v}$	: speed of board carrier
$\boldsymbol{V}$	: set of vertices
$\boldsymbol{x_{ip}}$	: binary variable which takes the value of 1 if node *i* is visited in position *p*.
$\boldsymbol{w_{ijp}}$:	: binary variable which takes the value of 1 if node *j* is visited in p^{th} position after node *i* is visited in $(p\text{-}1)^{th}$ position
$\boldsymbol{y_{tr}}$	: binary variable which takes the value of 1 if component type *t* is assigned to feeder slot *r*.

ABBREVIATIONS

AAT	**:**Acyclic Assembly Time
ACO	**:**Ant Colony Optimization
AFPP	**:**Adjust First Point Procedure
AFPTCP	**:**Adjust First Point according to Total Cost Procedure
AIP	**:**Arbitrary Insertion Point
AMP	**:**Adaptive Memory Procedure
ATMA	**:**Assembly Time Minimization Algorithm
ATSP	**:**Asymmetric TSP
BATA	**:**Back-Tracking Adaptive Threshold Accepting
BC	:Board Carrier
BSH	**:**Board Sequencing Heuristic
CPS	**:**Component Placement System
CR	**:**Component Retrieval
CVRP	**:**Capacitated VRP
EP	**:**Evolutionary Programming
EPD	**:**Extended Postpone Deviant
FAP	**:**Feeder Arrangement Procedure
FC	**:**Feeder Configuration
FCL	**:**Feeder Configuration List
FM	:Feeder Magazine
FSA	**:**Feeder Slot Allocation
FSMVRP	**:**Fleet Size and Mix Vehicle Routing Problem
GA	**:**Genetic Algorithms
GI	**:**Group Insertion
HGA	**:**Hybrid Genetic Algorithm
iATMA	**:**Inverse Assembly Time Minimization Algorithm
II	**:**Individual Insertion
IP	**:**Integer Programming
ISP	**:**Iterated Swap Procedure

LAP	**:**Linear Assignment Problem
LB	**:**Lower Bound
LBTA	**:**List Based Threshold Accepting
LP	**:**Linear Programming
MBTD	**:**Moving Board with Time Delay
MDVRP	**:**Multiple Depot VRP
MPCBA	**:**Multi-type PCB Assembly
NLP	**:**Nonlinear Programming
NNH	**:**Nearest Neighbor Heuristic
OAMP	**:**Optimal Assembly Mode Problem
PAP	**:**Pick and Place
PCB	**:**Printed Circuit Board
PD	**:**Postpone Deviant
PEP	**:**Pair-wise Exchange Procedure
PS	**:**Placement Sequence
PVRP	**:**Periodic VRP
RRT	**:**Record-to-Record Travel
RRTLEM	**:**Record-to-Record Travel with Local Exchange Moves
QAP	**:**Quadratic Assignment Problem
SA	**:**Simulated Annealing
SDTSP	**:**Sequence Dependent Traveling Salesman Problem
SDVRP	**:**Split Delivery VRP
SMT	**:**Surface Mount Technology
SPP	**:**Set Partitioning Problem
SVRP	**:**Stochastic VRP
TDTSP	**:**Time Dependent TSP
TS	**:**Tabu Search
TSP	**:**Traveling Salesman Problem
VFM	**:**Vehicle Fleet Mix
VRP	**:**Vehicle Routing Problem
VRPHE	**:**VRP with Heterogeneous Fleet of Vehicles
VRPPD	**:**VRP with Pick-Up and Delivery
VRPTW	**:**VRP with Time Windows

FIGURES

TABLES

CHAPTER I

INTRODUCTION

I.1. BACKGROUND AND MOTIVATION

The use of automated placement machines has arised some optimization issues which have attracted the interest of researchers for several decades. The automated placement machines are mainly used by telephone, computer and TV set manufacturers to populate (assemble) electronic components on a printed circuit board (PCB). A PCB is usually a rectangular plastic board on which the electrical circuit to be used in a particular electronic equipment is printed and the locations of the electronic components to be mounted are identified. It is possible to see the use of placement machines in any facility manufacturing electronic equipments. These placement machines may insert various electronic components to the predesigned holes on the PCB (which is called pin-thru insertion technology), or attach the components to the PCB surface precovered by an adhesive material (which is called surface mount technology, SMT). SMT has almost replaced pin-through insertion technology for PCB assembly and has enabled the production of high density (allowing many components to be placed onto a PCB in a small area) PCBs (Jeevan et al., 2002).

Most electronic products contain printed circuit boards (PCBs) as important components. PCBs are used extensively in a variety of products such as: computers, calculators, robots, remote controllers, business telephones, cellular phones, and many electronic instruments. In fact, the PCB market currently generates worldwide annual sales of $31 billion in 2007 and is estimated to be about $47 billion in 2012 (Drysdale, 2008).

Because a PCB may contain hundreds of electronic components in diverse shapes and sizes mounted at specific locations on the board, the assembly of PCBs is a complex job. To be more competitive in today's global marketplace, PCB assembly manufacturers are motivated to respond to emerging trends including high quality, low-cost and just in-time delivery. Therefore, in order to enhance their competitiveness, many PCB assembly manufacturers are dedicated to develop a computer integrated manufacturing system that is capable of producing an effective planning, scheduling and control procedure. Thus, PCB production has turned out to a highly automated activity over the last 20 years (Crama et al., 1997)

Generally, an SMT assembly line involves solder paste, component placement and solder reflow operations (a soldering process to adhere components on the PCB). A placement machine is very expensive (US$300,000 to US$1,000,000) and yet the SMT lines are typically designed such that the placement machine is the limiting resource or “bottleneck”, which is the key issue for assembly line optimization (Csaszar et al., 2000; Tirpak et al., 2000). For example, to produce 100,000 boards, a reduction of three seconds in placing the components, will save 5,000 minutes, which is, over 80 working hours. Thus, any additional gain in production capacity is considered valuable, mainly due to the cost of the SMT equipment. Consequently, for manufacturers to remain competitive in the growing PCB market, they must concentrate their efforts on improving the efficiency of their SMT assembly lines, especially on placement machines.

Typically, the placement operation begins by loading the PCB onto the placement machine (e.g. via a conveyer system). Then, the components are assembled onto the PCB guided by the optimization software that has been installed in the placement machine. Finally, once completed, the PCB is moved out from the placement machine (Leu et al., 1993).

In practice, the placement machines are usually not equipped with efficient optimization software (Shih et al., 1996). Until now, the PCB machine vendors and software companies have not been capable of solving even a single machine problem efficiently (Magyar et al., 1999). This indicates the need for research in this area. As was pointed out above explicitly, a small percentage of improvement can bring huge economical benefits. Hence it is still worth to study on them. It is necessary to solve the optimization problems inherent to these machines to get the best performance from them.

Nowadays, there are many types of placement machines available, such as sequential pick-and-place, multi-head, dual-delivery, turret type, multi-station, concurrent pick-and-place, etc. Various types of placement machines have different characteristics and restrictions (Wang et al., 1999). Thus, the PCB production scheduling process is highly influenced by the type of SMT placement machine being used.

Basically, the operations of these machines yield four major problems (Duman, 1998). These are allocation of component types to machines, determination of board production sequence, allocation of component types to feeder cells (also called the feeder configuration problem) and determination of component placement sequence. In many other studies, this list is extended or shortened but the last two have great influence and hence importance in optimizing the PCB placement machines (Smed et al., 2000, Crama et al., 2002). All of these problems are interdependent, that is solution of one affects the other. Depending on the principles of the machine, some may be trivially solved while in most cases they yield NP-Complete problems. Hence a solution aiming to achieve the optimum in all problems simultaneously is very difficult to find, if not impossible.

Generally, the determination of component placement sequence is akin to TSP or its variants such as Precedence Constrained TSP (Duman and Or, 2004), or multiple TSP (Duman, 2007b; Duman, 2009). On the other hand, feeder configuration problem turns out to be Quadratic Assignment Problem (QAP) in machines having movable feeder magazine (Duman and Or, 2007). How the placement sequencing and feeder configuration problems turn out to be the referred NP-Complete problems is explained in the thesis.

I.2. SCOPE AND OBJECTIVES

As it is listed in the previous section, various optimization problems exist in the area of production planning for the assembly of PCBs. The dilemma is that all sub problems are tightly intertwined and are NP-Complete, and the question arises as to which one should be solved first. As a consequence, this research is more focused on optimizing the component placement sequence and feeder configuration problems by assuming that other optimization problems are solved. This thesis is concerned with optimizing the assembly of a single PCB type on a single placement machine.

Since there are various types of placement machines, all which have different characteristics and restrictions, allocation of component types to feeders and the component placement sequence are highly influenced by the type of placement machine being used. Therefore, a selection has to be made as to which type of placement machines the research should be conducted upon. We have decided to focus this research on optimizing the chip mounter and chip shooter placement machines because they are widely accepted as the latest technology high speed placement machines (Duman 2007a, Chyu and Chang, 2008, Ho and Ji, 2003, Ho and Ji 2007, Ong and Tan, 2002).

Before clearly understanding the operations of the chip shooter machine, the early stage of the research focuses on the placement machine having a rotational turret and stationary feeder magazine since the operational methods of the machine are easy to understand and the optimization factors of the machine are not as complicated as other machine types. This machine is TDK brand mode RX-5A, and we will refer this machine as "chip mounter" throughout the study following the name used by the manufacturer.

A chip mounter consists of three parts; a rotational turret, a stationary feeder magazine and a board carrier moving in *x-y* directions (Figure I.1). The placement operation is performed by the placement heads mounted on the rotating turret and the placement location is fixed. Each assembly head has several nozzles to pick up various sized components. The feeder magazine is placed behind the machine (indicated by *pickup zone* in the figure).

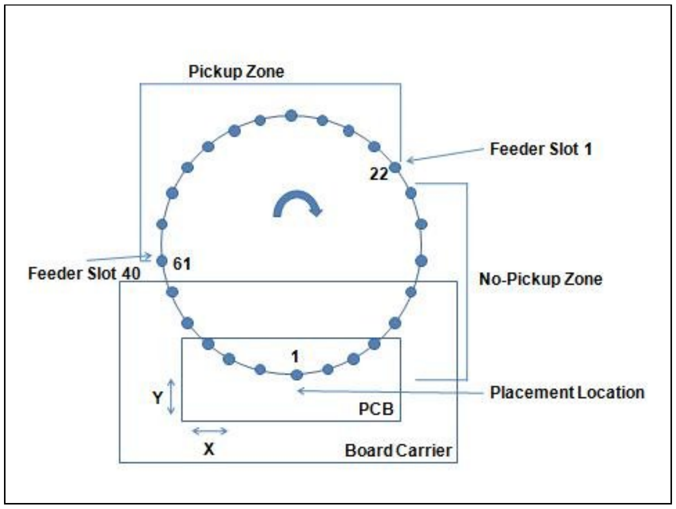


Figure I.1—TDK chip mounter placement machine

This machine has the feeder configuration and determination of placement sequence problems to be solved. It will be shown in the following sections that determination of placement sequence is formulated as the TSP or a generalization of it (such as SDTSP) while feeder configuration may be solved ideally by simple rules.

A chip shooter has a very similar structure with a chip mounter. It consists of three parts: a rotational turret, a linearly movable feeder magazine and a board carrier moving in *x-y* directions. The basic difference is that feeder magazine of chip shooter is movable. Also the number of placement heads is much less than the chip mounters. The placement operation is performed by the placement heads mounted on the rotating turret, that rotate between a fixed pickup and fixed placement locations (Figure I.2). Each assembly head has several nozzles to pick up components of various size. It can be observed that the actual benefit of chip shooters is their high speed because the pickup and placement operations are performed concurrently.

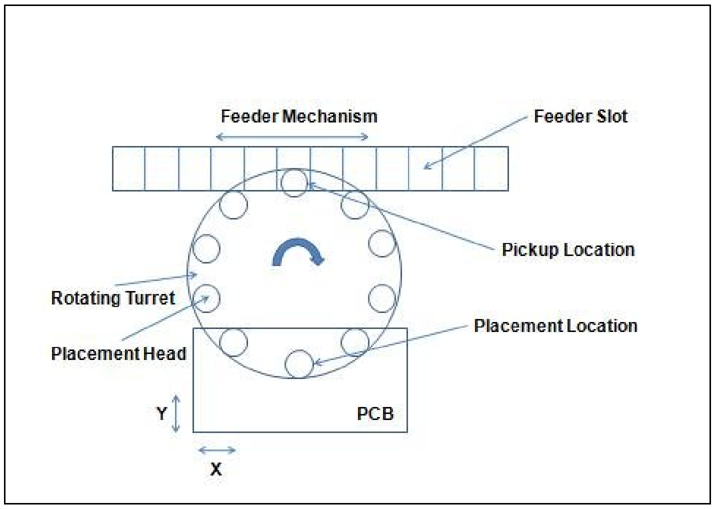


Figure I.2—Chip shooter machine

The feeder configuration problem turns out be the QAP and determination of placement sequence turns out to be the TSP or the Sequence Dependent TSP (SDTSP). However, in chip shooter machines, there is an interaction between these two problems and solution of one problem affects the solution of the other, because the pickup and placement operations are performed concurrently. Detailed explanation of the operations of chip shooter machines and how their problems are formulated are given in the following sections.

It must be noted that in both machine types, each assembly head on the rotational turret has several nozzles to pick up components types of different weights. When any of the heads picks up a heavier component type, the rotation speed of the turret is reduced. This phenomenon complicates the problem formulations because while determining placement sequence of components this complication must also be taken into consideration. The scope of this thesis also includes varying rotation speed values for the machine types undertaken.

The ultimate goal of this thesis is to propose effective techniques that can enhance the efficiency of chip mounter and chip shooter machines. To accomplish this, we identified the following objectives:

- Identify the placement machines in the PCB assembly industry that are similar to chip mounter and chip shooter machines.
- Bring out the mathematical programming formulations of problems emerging from the operations of chip mounter machines by considering varying rotation speed values of the rotational turret.
- Develop heuristics for optimizing the operations of chip mounter machines.
- Bring out the mathematical programming formulations of problems emerging from the operations of chip shooter machines by considering varying rotation speed values of the rotational turret.
- Develop a general methodology for optimizing the chip shooter machines with various characteristics.

I.3. METHODOLOGY AND CONTRIBUTIONS

The methodology taken up for this thesis is as follows.

Firstly, the problems emerging from the operations of chip mounter machines are identified and formulated. The formulation of these problems yield a new generalization of the TSP and thus a valuable contribution is made.

Then, the problems emerging from the operations of chip shooter machines are identified and formulated. The formulation of these problems also includes the above mentioned new TSP generalization.

Afterwards, a literature survey on the PCB assembly literature is performed and they are grouped according to the production cases, problems undertaken and solution approaches.

Several original heuristics are developed for chip mounter machines in order to obtain assembly times better than those given in the literature. These heuristics are also important contributions of this thesis.

The contributions of this study to the literature are as follows:

Journal paper:

- Alkaya, A.F.; Duman, E.; Eyler, M.A: "Assembly time minimization for an electronic component placement machine", *WSEAS Transactions on Computers*, 7, **(2008)** 326-340.

Conference proceedings:

- Alkaya, A.F.; Duman, E.: "Assembly Time Minimization of a Particular Placement Machine", *Proceedings of the 12th WSEAS International Conference on Applied Mathematics*, Cairo, Egypt. December 29-31, **(2007)**, 383-388.

- Alkaya, A.F.; Duman, E.: "Heuristics for a generalization of TSP in the context of PCB assembly", *Proceedings of the International Conference on Prospects for Research in Transport and Logistics on a Regional Global Perspective*, Istanbul, Türkiye, February 12-14, **(2009)**, 167-172.

- Alkaya, A.F.; Duman, E.: "A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines", *Proceedings of PICMET'09*, Portland, OR, USA, August 2-6, **(2009)**, 3296-3306.

Conference presentations:

- Alkaya, A.F.; Duman, E.: "Improving the efficiency of an electronic component placement machine", *Abstract Book of the 2nd International Conference on Control and Optimization with Industrial Applications*, Baku, Azerbaijan, June 2-4, **(2008)**.

- Duman, E.; Alkaya, A.F. “A New Generalization of the Traveling Salesman Problem” to be presented in *III. Congress of the Turkish World Mathematicians*, Almaty, Kazakhstan, June 30–July 04, **(2009)**.

I.4. OUTLINE

The rest of this study is organized in five chapters.

Chapter 2 gives a general background on PCB assembly, machines analyzed in this thesis, a survey of the studies on the analyzed machines and their optimization problems and a general overview of the related combinatorial optimization problems.

Chapter 3 includes the formulation of problems of the chip mounter machines. Several original heuristics developed for optimizing the operations of them are given.

Chapter 4 includes the formulation of problems of the chip shooter machines. Besides, a discussion on developing a general methodology for optimizing their operations is given.

Chapter 5 compares the performance of the proposed heuristics and the previous approaches given in literature and show the superiority of the proposed heuristics. The heuristics are tested on synthetically generated PCB data.

Chapter 6 gives a conclusion and main contributions of this thesis along with possible future work areas.

CHAPTER II

GENERAL BACKGROUND AND LITERATURE SURVEY

In this chapter, firstly a general background on PCB manufacturing systems is given in section II.1 and the focus of this thesis is explicitly stated. Then, in sections II.2 and II.3, an extended literature survey on the machine types and their problems that this thesis focuses on is given. Generally, it is better to use exact algorithms to solve the optimization problems to optimality. But, when optimal solutions are difficult to obtain, it is preferable to apply meta-heuristics to obtaining good solutions. Therefore, section II.4 covers the related combinatorial optimization problems and common solution approaches.

II.1. PRINTED CIRCUIT BOARD ASSEMBLY

The PCB assembly manufacturing systems are composed of several decision problems at different levels of difficulty. Smed et al.(2000) make a hierarchical classification scheme for the problems in PCB assembly on tactical and operational level. The classification is as follows: one PCB type and one machine (1-1), multiple PCB types and one machine (M-1), one PCB type and multiple machines (1-M) and multiple PCB types and multiple machines (M-M) (Smed et al., 2000). Using this scheme makes it easier to recognize the problems and to find suitable and efficient approaches for solving them.

For 1-1 category, the aim is to optimize the assembly time that a particular machine populates a given board instance. The main problems observed are, feeder configuration, placement sequencing and component retrieval. Feeder configuration is assigning components to feeder slots, placement sequencing is determining the order in which the components will be printed on the board and component retrieval is deciding from which feeder slot the head should retrieve the component if that component type is assigned more than one slot.

In 1-M category, several machines populate a PCB type in an assembly line and generally allocating component types to placement machines is handled as the main problem (Duman and Yıldırım, 2005). The objective is to balance the workload of the machines, hence eliminating bottlenecks.

In M-1 category, the observed phenomenon is population of many PCB types by one machine. In this case, component types for each board type may be diverse and minimizing feeder setup gains importance. So, two problems that arise are grouping PCBs into families and sequencing the production of PCBs.

In M-M category, the problems are in the form of scheduling problems. Assigning board types to assembly lines and grouping placement machines are the main problems observed.

A general framework that aims to minimize total manufacturing time considering many machines and many PCB types is almost impossible. Hence, all of the studies in PCB manufacturing industry focus on particular problems. These particular problems are of great interest for researchers because mostly they turn out to be generalizations of NP-Complete problems in combinatorial optimization.

There are many studies done on the PCB assembly machines in order to optimize their throughput. The studies in the literature regarding the PCB assembly optimization can be classified into several categories based on machine types, problem types, production environment, etc. Ayob and Kendall categorize the placement machines that are studied in the literature with their brief properties and solutions to problems emerging with them (Ayob and Kendall, 2008). They divide the machines into five categories according to their operation principles and moving parts. Crama et al. posed a survey from the classification of problems point of view (Crama et al., 2002). They define eight problems of PCB assembly process and give solution techniques developed for these problems so far. McGinnis et al. develop a framework for the PCB assembly process planning problems. The paper includes an overview of the essential elements of PCB terminology, assembly technologies and assembly system operations (McGinnis et al., 1992). In this thesis, it is not intended to give an exhaustive survey of the PCB assembly optimization research field. Hence, for general information about the PCB assembly process and different PCB assembly machines, we direct the reader to the references given above.

In this thesis study, the main interest is on the category of one PCB type and one machine (1-1) problems. Specifically, the analyzed machine types are the chip mounter and the chip shooter machines and problems arising from their usage. These problems must be solved concurrently because of their inherent design.

There are several studies that are concerned with optimization of either chip mounter or chip shooter machines. Many of these studies can be found in survey studies such as (Ayob and Kendall, 2008) and (Crama et al., 2002). However, there is no single resource that includes an extended survey on these machines and their optimization problems. This thesis includes an extended survey on the chip mounter and chip shooter machines, and optimization problems arising from their operations and thus fills this gap in the literature.

II.2. CHIP MOUNTER PLACEMENT MACHINE

In the literature, there is only one study performed by Duman (2007a) that analyzes the optimization of a chip mounter placement machine. In that study, after modeling the operations of the machine, two problems are formulated; the placement sequencing problem and feeder configuration problem. It is also shown that the placement sequencing problem can be modeled and solved as a classical TSP whereas the feeder configuration problem is solved in an ideal way by a simple procedure. The proposed algorithm is called Assembly Time Minimization Algorithm (ATMA). The results show that it brings an improvement in terms of total assembly time for this type of machines when compared with the solution approach used in practice.

To understand the problems emerging from the chip mounter machine, one should firstly investigate the design and working principles of the machine. In the following subsections, we give a summary of the study (Duman, 2007a) which also includes design and working principles of the machine.

II.2.1. Description of Operations

The chip mounter placement machine is usually encountered in assembly environments deploying SMT. The specific machine type is the TDK brand, model RX-5A placement machine which deploys SMT. Chip mounter has a rotational turret consisting of multiple heads which are responsible for the pickup of components from the component magazine and placing them on the PCB. The component magazine (also called the feeder mechanism) where the component tapes are installed is a circular structure behind the machine and it is stationary.

The rotational turret, which has 72 heads, collects the components to its heads (one to each head) in the placement order, from the component tapes placed in the component magazine, while rotating in clockwise direction (see Figure I.1). Even though the machine has actually 72 heads, a few are depicted in the figure for the sake of clarity. The head reaching the 'placement location' over the board moves down and makes the placement to the precise location on the PCB which is pre-aligned with the placement position by the board carrier actions. Thus, each head makes the placement exactly at the same Cartesian coordinate and the alignment of the placement point is made by the board carrier through its simultaneous and independent movements (leading to a Chebyshev distance measure) in the x–y plane.

These machines may handle component types of different weights (each head is equipped with three suction nozzles compatible with different weight categories). When any of the heads picks up a heavier component type, the rotation speed is reduced. For the particular machine considered, there are four discrete speed values corresponding to different component weight categories. These are 0.20, 0.23, 0.33 and 0.40 s per five degrees (or, per rotational movement of distance one head) ordered from lightest to heaviest components. On the other hand, the board carrier movement speed in both x and y directions is 120 mm per second which could practically be taken as constant (i.e. no acceleration or deceleration).

The component tapes are mounted to a component magazine behind the machine and this whole system is stationary, that is, it does not move in any direction. If the placement heads are numbered from 1 to72 in counter clockwise manner with number 1 being the placement head, up to 120 component tapes (depending on their width) can be installed to the feeder slots along heads 22 and 61 (pickup zone). The pickup times of the components from the magazine is completely determined by the order of their placement. Once the placement sequence is determined and the position of each component type in the magazine is entered to the computer, the component type that each head arriving at the placement location carries is known and fixed a priori. In other words, what component type should each and every head pick up is known. Then, during the rotational movement of the turret, the head(s) situating over the correct component tape move down and pick a component.

To summarize, a placement cycle of such a machine can be itemized as follows:

Item 1:

- turret rotates and the next placement head comes over the PCB
- board carrier aligns the new placement point under the placement head
- placement heads rotate if necessary to align the suction nozzle carrying the component to be placed

Item 2:

- the placement head moves down, makes the placement and moves up
- heads in the pickup zone that are above the appropriate component tapes move down and pick up components.

In the above cycle, the actions under each item are performed concurrently, and there is a sequential order between items one and two. The definition of the PCB assembly time minimization problem arising from the use of this kind of machines is discussed in the next section.

II.2.2. Problem Definitions

If the minimization of the PCB assembly time is defined as the master problem, the placement sequence and the feeder configuration are the two determinants of the assembly time. In other words, the assembly time of a PCB can be calculated as a function of the placement sequence and the feeder configuration.

Actually, what makes the master problem complicated is the varying rotational speed of the turret due to the different component weights. For the moment assume that all components to be populated on the PCB are at the same weight category (i.e. the turret has a uniform rotational speed throughout the whole assembly process). In this case, the master problem becomes equivalent to the placement sequencing problem (discussed below) and can exactly be modeled as a TSP with Chebyshev distance measure. There is no need to put the mathematical formulation of the TSP here, however, the distance between two placement points needs to be clearly defined. Since Chebyshev measure is used in the considered machine, the distance between two points i and j, $d(i, j)$, is defined as $\max\{| i_x - j_x |, | i_y - j_y |\}$, where i_x and i_y

represent the x and y-coordinates of point *i*, respectively. Cost is calculated in time units, hence it is defined as follows.

$$CF1(i,j) = \frac{d(i,j)}{v} \qquad \text{(II.1)}$$

where *v* is a predefined speed value.

If t_{ij} is defined as the time between the completion of consecutive placements at points *i* and *j*, then it can be calculated by using the following definitions.

t^0= component placement time (including placement head moving down and up time),

tt = turret time (turret rotation time required for the next placement head to arrive over the PCB),

Then, t_{ij} is given by the following expression

$$t_{ij} = t^0 + \max\left(CF1\left(i, j\right), \ tt\right) \qquad \text{(II.2)}$$

In many cases, the component placement time denoted by t^0 is small (in our case it is 0.05 s) and independent of the component placement sequence, so it may be dropped from the above expression without loss of generality. The turret time denoted by *tt* is also known as the free time since during this time interval the board carrier is free to move to any placement point without increasing the cost. In other words, the points at a Chebyshev distance of free time or less can be taken as equidistant points. If all components are at the same weight category, then there would be no feeder configuration problem. That is, the positioning of the component types in the magazine would not be an issue, since they would be arriving at the placement location in the order of placement sequence regardless of their actual positions. In fact, for the assembly of a single PCB, the feeder configuration could be important, since it influences the arrival time of the first component to the placement location. However, in mass production environments, the components arrive at the placement location continuously and this duration is certainly negligible in the total production time.

Consequently, if *tt* is constant, the master and the placement sequencing problems become equivalent and the TSP formulation given above defines both problems. However, in our case *tt* can take four possible values where its value is determined by the heaviest component type being carried by one of the heads of the turret. In other words, right after the placement of the component on head number 1 (see Figure I.1), the rotational speed value corresponding to the heaviest component being carried by the heads numbered 2–61 will be the current *tt* value for the next rotational step of the turret. On the other hand, the type of components being carried by heads 2–61, depends both on the placement sequence and the positioning of these component types in the component magazine. This means that the classical TSP formulation given for the PCB assembly time minimization problem cannot be valid anymore since it needs a constant distance between every pair of points. Furthermore, two sub-problems, namely the placement sequencing and feeder configuration problems arise.

Given a feeder configuration, the objective of the placement sequencing problem is to find a placement sequence of the components so that the assembly process is completed in the shortest time possible. Similar to the master problem, the time between two consecutive placements is determined by t_{ij}; however in this case, *tt* plays an important role in the determination of t_{ij}. The dependency of *tt* to feeder configuration makes the placement sequencing problem dependent on the feeder configuration problem. On the other hand, as explained above, in calculating the value of *tt*, the information of ‘if *j* is placed after *i*’ is not sufficient and in addition of *j* we had to know the next 59 components to be placed. In such a case, the basic requirement of ‘constant distances between pairs of cities’ is violated and thus the placement sequencing problem cannot be modeled as a TSP, turning out to be a more difficult problem even to formulate.

Actually, this problem turns out to be a generalization of TSP which we call as Sequence Dependent TSP (SDTSP) in which cost of travel between two points is based not only distance between them but also on the position in the sequence of the tour. We show and discuss how the arising problem turns out to be SDTSP in the following chapter.

However, as discussed in detail in the next section, for the parameters of the particular machine undertaken, mixing heavy and light components in the placement sequence is quite inefficient which, given the feeder configuration, allows us able to formulate separate TSPs for the groups of components in different weight categories.

The objective of the feeder configuration problem is to find the optimum positioning of the component tapes within the magazine so that, given a placement sequence, the number of slower steps taken by the turret is minimized. As the *tt* values of our machine are 0.40, 0.33, 0.23 and 0.20 s (they will be denoted by tt_4, tt_3, tt_2 and tt_1, respectively), a good feeder configuration should first minimize the steps taken in 0.40 s and then the steps taken in 0.33 s and 0.23 s. This can be stated as a goal programming model:

min z_1=steps taken in 0.40 s

min z_2=steps taken in 0.33 s

min z_3=steps taken in 0.23 s

s.t. all component types are assigned to a feeder slot

As discussed in the next section, the above modeling approach implicitly requires the allocation of heavier component types closer to the PCB in order to make their discharging at the placement location quickly after a short travel. Note that, the instant when to pick up a heavy component is a consequence of the placement sequence and thus, the feeder configuration problem is highly related to the placement sequencing problem.

A simple and effective solution methodology for the placement sequencing, feeder configuration and the master problems proposed by (Duman, 2007a) are described in the following subsection.

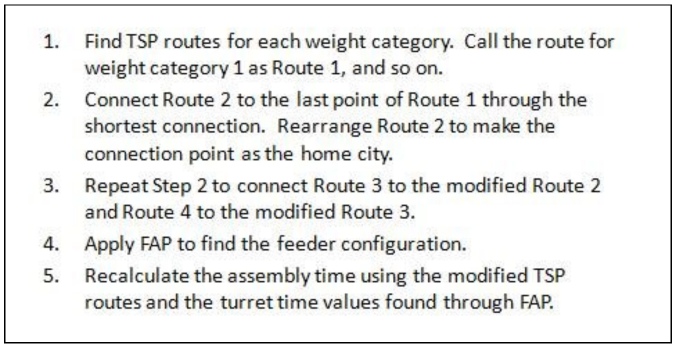
II.2.3. Solution Methodology

Between the two sub-problems, the placement sequencing problem (for a given feeder configuration) can be regarded as the main problem since its solution directly gives the PCB assembly time. Accordingly, the feeder configuration problem should be regarded as the auxiliary problem.

II.2.3.1. Assembly Time Minimization Algorithm (ATMA)

Recall that, the variable and complicated nature of turret time not only makes the placement sequencing problem difficult but also makes the TSP formulation infeasible. However, it turns out that, for the placement machine investigated, this change in the tt_k values makes placement sequences of mixed light and heavy components quite inefficient. Accordingly, it seems to be a good idea to place all of the lighter components first and then the heavier ones (or, vice versa). This way, the tt_k values corresponding to each weight category would be constant and it would be possible to use the TSP formulation to find the placement sequences within each weight category. This is the idea behind the Assembly Time Minimization Algorithm (ATMA) that he suggests for the solution of the master problem. After finding the TSP routes for each weight category, it will be necessary to connect these routes in a cost-efficient way and to arrange the feeder according to the resulting placement sequence. For finding a good feeder configuration, he developed a simple and effective procedure, named the Feeder Arrangement Procedure (FAP) and it is discussed in the following sub-section.

The steps of ATMA are given in Figure II.1.



1. Find TSP routes for each weight category. Call the route for weight category 1 as Route 1, and so on.
2. Connect Route 2 to the last point of Route 1 through the shortest connection. Rearrange Route 2 to make the connection point as the home city.
3. Repeat Step 2 to connect Route 3 to the modified Route 2 and Route 4 to the modified Route 3.
4. Apply FAP to find the feeder configuration.
5. Recalculate the assembly time using the modified TSP routes and the turret time values found through FAP.

Figure II.1—Algorithm ATMA

For the solution of the TSPs in step1, one of the heuristic methods performing well under the Chebyshev distance measure can be utilized (Bozer et al, 1990). He suggests using Convex–Hull and Or-Opt algorithms for the solution of the TSP. These algorithms are effectively used to solve the TSP in this context and very successful results were obtained (Duman, 1998; Duman and Or, 2004). The Convex–Hull algorithm is proposed by (Or, 1976) and (Stewart, 1977), whereas Or-Opt (Or, 1976) is an improvement procedure applied to the result of the Convex–Hull algorithm.

The solution of the feeder configuration problem is discussed next.

II.2.3.2. Feeder arrangement procedure (FAP)

The philosophy of FAP is based on the answers of the questions: Should components in a weight category be placed in the magazine as a group next to each other and if yes what part of the magazine should be allocated to each group and what should be the internal arrangement within each group? The arguments used to answer these questions are given below:

Recall that the pickup device slows down as soon as a heavy component type is picked up and it goes in slow speed until all heavy components are discharged at the placement point (location of head number 1 in Figure I.1). In order not to cause multiple, separate slowdowns for each heavy component pickup, it is better to pick up heavy components to consecutive heads of the turret. For this to happen, the heavy components (and the light ones also) should be placed as a group in the magazine.

Next, we need to decide what part of the magazine should be allocated to each group. The idea is, once a heavy component is picked up, it should be discharged as soon as possible. This is possible if the tapes of the heavy components are located closer to the PCB with respect to the rotation direction of the turret. In other words, if the feeder slots are numbered in a counter clockwise manner with number 1 being closest to the PCB (see Figure I.1), the first n_4 (n_i denotes the number of component types in weight category i) slots should be assigned to group-4, next n_3 slots to group-3, etc.

To further clarify the above discussion, assume that 60 components (N=60) are to be populated on a PCB, where 50 of them are in group-1 (N_1=50), 10 of them are in group-2 (N_2=10) and n_2=10 (i.e. only one of each group-2 components is to be placed). In this case, if on the contrary to the above discussion, group-2 components are located to the, say, last 10 slots of the magazine, the turret time value would be 0.23 s applicable to each and every component placement. That is, the turret time value of 0.20 s corresponding to group-1 components will have no use. This is because, before being able to place the first group-1 component, a component from group-2 will be picked up and the turret will slow down. However, if they are located in the first 10 slots in the magazine then, it is possible that, the turret time value will be 0.20 seconds for the first 30 component placements and 0.23 seconds for the next 30 (he names any feeder configuration making this possible an ideal one). To see why this is so, recall that a component picked up from slot 1 can arrive at the placement location after 20 steps of the turret and thus, after the placement of 30 group-1 components, the first group-2 component is picked up from slot 1 (ideally) and the turret slows down. This (30, 30) distribution between the groups will also be the same for the following PCBs to be populated since in mass production the components are picked up at heads continuously.

As another example, assume that 100 components where N_1=80, N_2=10, N_3=5 and N_4=5 are to be populated on a PCB with n_2=10, n_3=5 and n_4=5. According to the above discussion, first five slots will be assigned to group-4, next five slots will be assigned to group-3, etc. In this case, in the continuous assembly process of the PCB the tt_k values of 0.20, 0.23, 0.33 and 0.40 s will be used 50, 15, 10, 25 times, respectively. In other words, under an ideal feeder configuration, the number of rotational steps of the turret in a speed category i (SC_i) can be found using the following formulas:

$$SC_1 = \max\{0, (N_1 - (20 + n_4 + n_3))\} \tag{II.3}$$

$$SC_2 = \max\{0, (N_1 + N_2 + SC_1 - (20 + n_4))\} \tag{II.4}$$

$$SC_3 = \max\{0,(N_1 + N_2 + N_3 - SC_1 - SC_2 - 20)\} \tag{II.5}$$

$$SC_4 = N - SC_1 - SC_2 - SC_3 \tag{II.6}$$

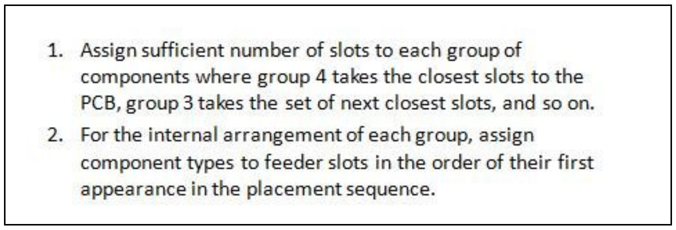
The above formulas are valid if there are components to be populated in every weight category. Otherwise, if there are no components in some weight categories, the weight categories should be numbered from one to K (K being the number of different weight categories in the problem and K<4), the indices for the remaining (4-K) categories should be dropped from the formulas and first (K-1) of them should be used for the first (K-1) groups. To find the number of rotational steps in the speed category corresponding to the heaviest group (i.e. group K), the formula given for SC_4 should be used (e.g. consider the previous example which is composed of 60 components).

Now, he discusses how to determine the internal arrangement of component types within each category. In the following discussion, only two groups of components (group-1 and group-2) are assumed, for simplicity. The methodology presented can then easily be extended to four groups of components.

The proposed idea in determining the internal arrangement of group-2 components, is not to pick up any of them earlier than necessary. For example, assume that there are five group-2 component types and are denoted by the letters *A,* *B, C, D* and *E*. Let the placement sequence be *A-C-E-C-A-D-D-B* (note that, N_2=8 and n_2=5). In this case, it is less desirable, for instance, to place component type C to feeder slot 5, since it will be picked up quite early. The idea is, since it is the second component type in the sequence, it should not be assigned to a feeder slot number greater than 2, or otherwise, it will be picked up early. Since, this is true also for the other component types, a good solution is to determine the feeder configuration in the order of the first appearance of component types in the placement sequence. For the above example, such a good feeder configuration is (*A,C,E,D,B*). This solution is ideal in that the turret does not slow down earlier than necessary. In other words, the number of the slow steps of the turret will be 28 (20+8).

Under the above feeder arrangement for the group-2 components, at the first step of the turret one of component types *A, C* and *E* will be picked up by heads numbered 22, 23 and 24, respectively (according to the numbering of heads in Figure I.1). Then the turret rotates one step, but this time no group-2 component is picked up. After one more step, head 25 picks a component type *C* and head 27 picks a component type *D*. In the next step, head 28 picks up a component type *D* and head 29 picks up a component type *B*. Finally, in the fifth step head 26 picks a component type *A* and this way the placement sequence is maintained on the consecutive placement heads.

Note that, for some placement sequences, more than one configuration may perform equally well. For example, in the case of the placement sequence *A-A-C-E-C-A-D-D-B*, both of the configurations (*A,C,E,D,B*) and (*A,E,C,D,B*) perform equally well (number of slow steps of the turret are the same for both solutions) and both are ideal. In case of the existence of more than two weight groups, the above internal arrangement procedure can be applied to other weight categories as well. The FAP is given in Figure II.2.



1. Assign sufficient number of slots to each group of components where group 4 takes the closest slots to the PCB, group 3 takes the set of next closest slots, and so on.
2. For the internal arrangement of each group, assign component types to feeder slots in the order of their first appearance in the placement sequence.

Figure II.2—Procedure FAP

After the arrangement of the component tapes in the magazine is determined using the FAP procedure, as pointed out by the last step of ATMA, the cost of the total placement cycle should be calculated again using the correct tt_k values.

In the following section, a general background on some related combinatorial optimization problems and their solution techniques will be given.

II.3. CHIP SHOOTER PLACEMENT MACHINE

There are many studies performed for optimizing the operations of chip shooter machines. Before giving the details of an extended survey on chip shooters, it will be good to give the details of its working principles and problems emerging from its operations.

There are various models of chip shooters with different operation parameters. But at the basic, chip shooters consist of three parts (see Figure I.2):

- A board carrier: The PCB lies on this carrier which is able to move horizontally and vertically in a concurrent manner. To achieve this concurrency, the carrier is controlled by two independent motors.
- A feeder carriage (rack): The feeder carriage consists of the so-called slots. It is linearly (horizontally) movable. A feeder is a reel that stores components of a single type, and is attached to a slot on the feeder carriage.
- A rotational turret (carousel): This is a device which is able to transport components from the feeder carriage to the board by rotating clockwise. The turret contains an even number of placement heads (H) that are arranged on the perimeter of the turret.

In the following subsections, firstly, the operation principles and problems emerging from the operations of chip shooter machines will be given in detail. Then, the varying turret rotation time of these machines and the assumptions used in the literature to cope with the particular machine specifications are summarised briefly. In the last subsection, a detailed survey of chip shooter literature is given.

II.3.1. Operation Principles

The placement operation is performed by the placement heads that rotate between a fixed pickup and fixed placement locations. Each assembly head has several nozzles to pick up components of various sizes. A large nozzle is used to pick up and place large components.

The function of the feeder carriage is to move in the x-direction to align the correct component feeder under the pickup head in the fixed pickup location.

The board carrier secures the PCB and positions it under the placement head in the fixed placement location where the loaded component is mounted.

The movements of these three parts are almost independent of one another (the slight interdependency will be explained later). Since individual activities of a chip shooter machine all happen simultaneously, the resulting model of its movement is very complex. But by itemizing the movements of the machine, we can extract the modus operandi of the placement machine.

1. turret rotates (indexes) one step (i.e. rotates $360/H$ degrees)
2. board carrier aligns the new placement point under the placement head
3. placement heads rotate if necessary to align the suction nozzle carrying the component to be placed
4. the feeder carriage moves to place the correct component feeder under the pickup head
5. the placement head makes the placement
6. the pickup head retrieves the component

Operations 1-4 are performed concurrently which is followed by the concurrent movement of items 5-6. Hence, there is sequential order between these two groups of operations. These movements are repeated for each component of the PCB.

The time for retrieval and placement (operations 5 and 6) is normally assumed to be equal (t^0) and the same for all components and no optimization of this time is possible (Chyu and Chang, 2008). Therefore, after multiplying it with number of components (N), it is added as a constant to the assembly time formulations.

On the other hand, as there are three moving parts in a high speed chip shooter (board carrier, feeder carriage and carousel), each moving part has to wait for other two parts to complete their movements before the next component can be picked up or placed. Hence, the mechanism that takes the longest to complete the operation dictates the placement rate during each pickup and placement. Mathematically, the placement time for any component is the maximum of three movement times (turret rotation time, feeder carriage movement time and board carrier movement time) plus the constant time for retrieval and placement. The operation time for aligning the suction nuzzle carrying the component to be placed is mostly very smaller than the indexing of the turret, hence does not contribute to the cost calculation.

The reader should easily see that the component gripping sequence is *H/2* indices/heads ahead of the component placement sequence. Also, it can be observed that the actual benefit of chip shooters is its high speed because the pickup and placement operations are performed simultaneously.

This machine type is usually called a chip shooter machine but some researchers choose to use the term concurrent chip placement machine (Yeo et al., 1996, McGinnis et al., 1992, Ammons et al., 1993).

Examples of common chip shooter machines include the Fuji CP4, CP4-2, and CP4-3 machines, which have 12 mounting heads and 160 feeder slots, and the Fuji CP6 machine, which has 20 placement heads and 160 feeder slots (Ellis et al., 2001). In Figure I.2, we illustrate a chip shooter machine with 10 placement heads and 12 feeder slots. Chyu and Chang talk about CP 642 and CP 742, which are more recent chip shooter models (Chyu and Chang, 2008). Another study is on CM82 chip shooter machine with 18 placement heads and 100 feeder slots (Ohno et al, 1999).

II.3.2. Problems emerging from the operations of chip shooter machines

Given a PCB type, calculation of assembly time is based on the placement sequence of components and assignments of feeders to slots. Hence, there are at least two problems that await optimization in order to obtain a good assembly time. Once given a feeder configuration, determination of component placement sequence (placement sequencing problem) recalls the well-known Travelling Salesman Problem (TSP) since a tour is developed for the component placements to minimize the placement time (McGinnis et al., 1992, Ammons et al., 1993).

On the other hand, once given a placement sequence, determination of assignments to slots (feeder configuration) is similar to a Quadratic Assignment Problem (QAP), since the feeders are assigned to slots on the feeder carriage and the cost of the assignment is affected by the location of the other feeders (McGinnis et al., 1992, Ammons et al., 1993).

For chip shooter machines, the placement sequencing and feeder configuration problems are highly interdependent. Feeder configuration influences the placement sequence since the time spent between two consecutive placements depends on the feeding time (thus, the feeder configuration) of the next component to be placed to the head. On the other hand, placement sequence influences the feeder configuration, since the amount (time) of linear feeder carriage movement depends on the distance between the feeder locations of the consecutively placed components (placement sequence).

Both TSP and QAP are NP-Complete problems for which general algorithm for solving all instances are not available. Thus, optimizing feeder configuration and placement sequencing problems simultaneously becomes a more challenging problem.

There may be other problems for optimization in chip shooter machines. For example, when a component type is assigned more than one slot on the feeder carriage, component retrieval problem arises during the assembly of PCBs. Component Retrieval Problem can be defined as follows: for a given placement sequence of components on the board, and for a given assignment of component types to (possibly multiple) slots of the feeder carriage, decide from which feeder slot each component should be retrieved.

II.3.3. Turret rotation time

Chip shooter machines have an operating principle that is fundamental for obtaining a realistic assembly time and gives way to new optimization opportunities. This is the varying speed of the turret according to the transferred components. This property of the machines is mostly overlooked by researchers.

The rotation of the turret is performed at high speed. Hence, substantial centrifugal forces work on the components transferred. These forces cause the component to be dislocated or even to be lost. Their strength depends on various characteristics of the components such as size, weight and type of packaging. Thus the control unit of chip shooters determines the possible rotation speed depending on the individual component type. For this purpose, all component types are assigned to rotation speed classes which allow only a comparatively slow transfer. The feasible rotation speed does not only depend on the component that is being placed next or the component that has just been picked up but on the entire set of components that is transferred in the turret at that moment (Grunow et al., 2000).

Hence, a grouping of component types according to their turret rotation speed is necessary. If this is not considered, the assembly time of each board will increase because transferring one heavy component among light components will decrease the turret speed for *H/2* steps.

In the literature, there are very few studies taking the varying turret rotation speed into consideration (Ellis et al., 2001, Chyu and Chang, 2008, Grunow et al., 2000). Crama et al. (1997) only remind the reader about the change of turret rotation speed depending on the carried components but they neglect this property while modeling the machine in order to obtain simplicity and by arguing that this property of the machine will have only marginal effects to the assembly times, which we believe is a wrong argument as explained above (Crama et al., 1997).

II.3.4. Assumptions for machine specifications and free time values used in literature

This subsection gives a brief overview of assumptions for machine specifications and free time values used in the previous studies on chip shooters. However, they can only be explained by using some notation. Below we will give the meaning of necessary notation for modeling the chip shooter machine that is used in the following subsections. An extended notation will be provided in following chapter.

H: number of heads on the turret

N: total number of components to be placed,

n: number of component types or number of feeder slots (we assume that one component type is assigned to only one slot)

K: number of weight categories

t^0: time required to retrieve a component from the feeder magazine and concurrently populate a component on the PCB after all the mechanisms are positioned,

tt_k : turret rotation time for each weight category k, (When $k=1$, the turret rotation speed is the maximum, that is, turret rotation time is minimum) (tt is used if weight category among components is disregarded)

$C2_{rs}$: time required for the feeder carriage to travel from feeder r to feeder s

Bard et al. and Crama et al. discuss the so-called free movement of the component placement operation (Bard et al., 1994, Crama et al., 1997). Between any two consecutive grips or placement operations, the turret must rotate one station, which takes a certain amount of time. A movement of feeder carriage or board carrier is free if it takes less time than that of one turret rotation step (Chyu and Chang, 2008).

The value of a free feeder carriage movement is taken as no more than one slot for Fuji CP-IV and Fuji CP4-3 machine in (Crama et al.,1997) and (Klomp et al., 2000). The operational time estimators provided by Ellis et al. for Fuji CP4-3 indicate that the free movement for feeder carriage is zero when turret rotation uses maximum speed setting (Ellis et al., 2001). However, this free movement will increase to one or more slots as turret speed is set to a slower value.

The parameters extracted in (Ellis et al., 2001) are also used in (Chyu and Chang, 2008). We know that, since the turret must rotate one station for each pickup and placement, there is an *u*-slot free movement for the feeder carriage if $C2_{r,r+u} \leq tt_k$, where tt_k represents the time of rotating the turret one station with the speed setting indexed k. When k=1, the turret rotation speed is the maximum (turret rotation time is minimum). Their findings about Fuji CP4-3 machine can be summarized as follows. At the maximum speed the velocity of the feeder carriage is slower than the rate of movement for both turret rotation and the board carrier. The speed setting for the turret rotation is based on the accuracy of component placement. In practice, such speed settings are not more than $C2_{r,r+1}$, i.e. moving feeder carriage only one slot. For components of larger sizes or requiring stringent accuracy (weight category k), the time for one-station turret rotation is set between $C2_{r,r+1}$ and $C2_{r,r+2}$, i.e. $C2_{r,r+1} < tt_k < C2_{r,r+2}$. As component size decreases, the turret speed approaches the maximum. According to Ellis et al., $C2_{r,r+1} =\sim 1.25tt_1$ and $C2_{r,r+2} =\sim 1.5\ tt_1$ (Ellis et al., 2001). Moreover, $C2_{r,r+1}$ and $C2_{r,r+2}$ are approximately equal to 4.0 cm and 6.0 cm moving distances of board carrier (Chyu and Chang, 2008). After these findings, Chyu and Chang developed an algorithm such that in the case of $tt_k < C2_{r,r+1}$, the component placement sequence is determined on a slot-by-slot basis that follows the order of the feeder arrangement (Chyu and Chang, 2008). In the case of $C2_{r,r+1} < tt_k < C2_{r,r+2}$, there is a one-slot free movement of feeder carriage, and thus the component placement sequence is determined based on components in two neighboring slots.

Fuji CP4-3 machine is also analyzed in (Klomp et al., 2000). However, they did not take into account the varying turret rotation time values in their study. After defining the free time, they noted that in each rotation of the turret, there is a one slot free movement for feeder carriage and approximately 10 cm for board carrier.

After analyzing the operations of Fuji IV placement machine, Yeo et al. (1996) used the following information about the machine (Yeo et al., 1996). Travelling distance between placement operations should be at most 20 mm (in Chebyshev metric) and the free time of the feeder movement should be at most one slot. They did not consider about the varying turret rotation time values.

The operations of Fuji FCP-IV chip shooter machine are also analyzed in (Kumar and Luo, 2003). The basic observation about the machine is that feeder magazine has a much slower movement when compared with PCB board carrier.

It is observed in (Crama et al., 1997) that, free movements of the feeder carriage correspond to one position on the rack, that is, repositioning the feeder carriage by zero or one slot is free. Concerning the table, free movement corresponds to approximately two cm on the table containing the PCB. They conclude that since the average PCB is approximately 20×30 cm, and the feeder rack contains mostly about 100 feeders, long table movements are less time consuming than long feeder rack movements.

In the machine analyzed in (Ng, 1998) and (Ng, 2000), (without giving the model) due mainly to acceleration, the time of moving one or two slots is practically the same and is denoted by α. α is an empirical constant greater than tt where tt denotes the constant time for rotating $\pi/5$ radians of rotational head (he considers a unique turret rotation time). The time required then increases linearly with the movement of a higher number of reel positions ($\alpha r/2$ for $r=3,4,\ldots,n$). Thus the time required for moving the feeder by one or two slots is often more significant than that of the movement of board carrier; and in their heuristic algorithm, the most important strategy is then that the movement of feeder carriage should be within one or two reel positions. Correspondingly the time required for the movement from one component location to another on the PCB should be limited to α. In short, Ng states that the machine that he analyzes has a feeder magazine speed that is slower than one step rotation of the turret (with 10 heads). Hence, he seeks the solution by not moving the feeder as much as possible and by moving the board carrier as much within the rotation speed. The solution proposed is a groping technique similar to proposed by (Kumar and Luo, 2003).

Chen and Chyu tried to solve PCB grouping, feeder configuration and placement sequencing problems for a chip shooter machine in M-1 production case (Chen and Chyu, 2002). The model of the machine is not spelled out but its operation parameters are summarized as follows. Board carrier movement rate is 0.0075s/mm, feeder magazine movement rate is 0.15s/slot, turret rotation rate 0.15s/index and rack capacity is 100. Number of heads is not given as a parameter but the illustration has 20 heads. From these values, we conclude that free movement of feeder magazine is one slot and board carrier is 20 mm.

In the literature, another discussion is about placement time for first and last few components on a PCB. In (Ellis et al., 2001) and (Vittes, 1999), it is argued that the placement time for the first component is assumed to be the fixed pick and place time (t^0) and for the last $H/2$ components the feeder carriage movement time is assumed to be zero since there are no more components to pick up.

Also, in (Chyu and Chang, 2008) and (Ong and Tan, 2002), it is assumed that the first six turret rotations only pick up components, and the movement time for each pickup is the greater time value between the rotation of turret station and the corresponding travel of the feeder carriage. For the last six turret rotations, which only perform placements, the movement time of each placement is the greater time value between rotating one station of the turret and the corresponding board carrier movement. As for the turret rotations in between, the movement time is the maximum time value of all three mechanism movements. However, these assumptions are erroneous when continuous production is considered. A more acceptable statement is that for the first few components assembled in a batch of PCBs, there are only pick-up movements and no placement movement. For the last few components of the same batch, there are only placement movements and no pick-up movement. Therefore, if the quantity of PCBs in a batch is very large, these boundary effects can be neglected (Leu et al., 1993), (Ho and Ji, 2003).

II.3.5. Survey of the literature on chip shooter machines

In this subsection, a detailed survey of the studies on optimization of chip shooter machines is given. Most of the studies about chip shooter machines (23 out of 29 papers considered here) assumed one PCB type and one machine (1-1) production case. More than half of these studies focus on solving both the feeder configuration (FC) and the placement sequencing (PS) problems, while some of them focus only on placement sequencing, feeder configuration or component retrieval (CR) problems separately. There are a few studies that also consider component retrieval problem together with placement sequencing and feeder configuration problems. However, it should be noted that in (Klomp et al., 2000), the authors arrive at an interesting conclusion that there is no advantage in assembly time reduction when feeder duplication is used. The rest considering M-1, 1-M and M-M production cases mostly examine PCB grouping and/or component allocation problems with or without considering FC, PS, CR problems.

As it is shown in previous sections, feeder configuration turns out to be QAP and placement sequencing turns out to be TSP (or its generalizations) in chip shooter machines. The operation of the machine makes these problems interdependent. However, there are other machine types whose operation principles also yield TSP and QAP and their interdependency. These machines are either from the PCB manufacturing industry (such as Panasert RH placement machines) or from other industries (such as numerically controlled punch press). Researchers have also investigated these machines and proposed improvement techniques. For obtaining a full survey of literature, summary of some of these studies (that address solving QAP and TSP simultaneously) are included.

Under the light of this short summary, the survey is divided into three parts; 1-1 production case, other production cases and solving FC and PS in other environments.

II.3.5.1. 1-1 Production environment

More than half of the studies in 1-1 production environment focus on solving FC and PS problems in a combined manner. These studies either use genetic algorithms (GA) or apply an iterative approach for solving the problems.

Genetic Algorithm approach

Two link chromosome structure is first proposed in (Leu et al..1993) and it is used by many authors. They present application of the genetic algorithm approach to solve printed circuit board assembly planning problems. They address three main types of placement machines including the chip shooters. The developed GA finds the placement sequence and feeder configuration simultaneously. The GA has two-links where the first link represents the assembly sequence whilst the second link represents the feeder arrangement. Four genetic operators were applied to each link: crossover, inversion, rotation and mutation. The algorithm allows setting of parameters that specify the number of solutions to be created at each iteration using each of the genetic operators. Total assembly time is used as a fitness function, with the aim being to minimize the assembly time. It is argued that the solution found was nearly optimal. The algorithm is tested for a PCB with 50 components. It was also shown that the method is easily adaptable to the planning problems for many types of assembly machines.

Three problems arising in PCB assembly environments are used for comparison of Simulated Annealing (SA), GA and Evolutionary Programming (EP) in (Nelson and Wille, 1995). These are TSP, pick-and-place (PAP) and moving board with time delay (MBTD) problems. PAP is observed in machine structures where the assembly head needs to pick up the components from the feeders and place them on the appropriate locations on the board. The environment where MBTD arises is chip shooter machines and in MBTD the optimization is done by finding minimum time assembly sequence and component assignment to feeders. The data used for comparison is taken from (Leu et al., 1993). As a result, the study emphasizes that SA has the potential to generate best solutions, if run for a sufficiently long time, but tends to converge most slowly of the three techniques investigated. Hence, in cases where the same type of PCB will be produced for long periods of time it might be valuable to use extensive SA runs to find the best possible solutions. On the other hand, in situations where the types of PCBs to be assembled change rapidly, EP may be the method of choice because less programmer intervention and fine tuning of parameters is needed for it.

Khoo and Loh addressed both feeder configuration and placement sequencing problems simultaneously (Khoo and Loh, 2000). They formulated the combined problem as a multi-objective optimization problem and utilized a GA to generate the placement sequences and feeder configurations. A framework of the prototype system and the derivation of the multi-objective function are described. The prototype system was validated using examples of (Leu et al., 1993) which are slightly improved.

Ong and Tan focus on the application of GA to solve the moving board with time delay problem, in particular to problems pertaining to high-speed turret-head chip shooters (Ong and Tan, 2002). Moving board with time delay problem is the combined problem of placement sequencing and feeder configuration. Defined in (Nelson and Wille, 1995), it is so called because there is usually a time delay caused by the motion of the feeder magazine or the turret indexing: the faster parts of the machine have to wait for the slowest moving element to complete its task. The formulation of objective function is based on producing one board type (not continuous production), that is for first and last six turret rotations the formula changes. A solution is represented using two chromosomes, the feeder configuration list (FCL) and placement sequence (PS). GA is implemented such that PS are mated first and then FCL are mated secondly. After each mate, four different offsprings are obtained and each one is compared with worst parent and is stored if it is better than that worst parent. In each iteration of GA, the fittest parent is selected and others are mated with this fittest one. The test data is obtained from (Leu et al., 1993) and it contains 50 components with 10 component types. The results obtained have shown improvements over those obtained from previous papers by (Leu et al., 1993) and (Nelson and Wille, 1995).

Ho and Ji developed a Hybrid GA (HGA) for chip shooter machines and applied it to solving the placement sequencing problem only in (Ho and Ji, 2005), to the placement sequencing and feeder configuration problems concurrently in (Ho and Ji, 2003), and to the placement sequencing, feeder configuration problems concurrently by also considering component retrieval problem in (Ho and Ji, 2006). However, they neglected to point out the model of the machine together with its operation details.

(Ho and Ji, 2003) studied on the chip shooter machine and they addressed to solve both feeder configuration problem and component placement sequence problems simultaneously. They use Genetic Algorithms (actually Hybrid GAs and compare the performance results with the ones obtained with GAs). In that study, it is claimed that no researcher has studied the component scheduling problem for the chip shooter machine by using HGAs. They call it HGA because they use different search heuristics such as nearest neighbor heuristic (NNH), 2-opt heuristic and an iterated swap procedure (ISP) mounted in crossover and mutation operations. The principle of the ISP is very similar to that of the 2-opt local search heuristic, except that some instead of all two swaps are examined. Their genetic algorithm represents a chromosome as two-link structures. As in (Leu et al., 1993), the first link represents the sequence of the component placement and the second link represents the feeder setup. The fitness function represents the total assembly time. The algorithm starts by setting the GA parameters such as population size, number of iterations, crossover rate and mutation rate. Then, initial chromosomes are created. For each chromosome, the first link is generated by the NNH while the second link is generated randomly. For improving initial chromosomes, the 2-opt local search heuristic is performed on the second link and the ISP is performed on the first link. After evaluating the fitness value for each chromosome, the initialization phase is over. Afterwards, the following actions are done sequentially for a number of times. Offsprings are generated by modified crossover heuristics mutation and inversion mutation operations. The 2-opt local search heuristic is performed on the second link and ISP is performed on the first link for each offspring generated. After evaluating all current chromosomes, a predetermined number of best chromosomes are selected for use in the next iteration.

The performance of the HGA was evaluated by the use of the PCB example of (Leu et al., 1993). Ho and Ji argued that the HGA in (Ho and Ji, 2003) is superior to a simple GA used by (Leu et al. 1993). They obtained better initial solutions, better final solutions with smaller population sizes and fewer iterations compared to (Leu et al., 1993).

In (Ho and Ji, 2006), a hybrid genetic algorithm is presented to solve the combined problem of the three subproblems (Component retrieval, Component sequencing, Feeder arrangement) simultaneously for the machine. A two-link representation together with the retrieval plan was used to solve the problem. When a component type is assigned to two feeders, a retrieval plan must be set up so as to determine which feeder a component should be retrieved from. The retrieval plan adopted in this paper is similar to the NNH. For the first component, if its type is stored in two feeders, then select a feeder randomly. However, if the types of the remaining components are stored in two feeders, then select the feeders as close as possible to the previous ones so that the movements of the feeder carrier are minimized.

Chyu and Chang propose a genetic-based algorithm to solve the placement sequencing and the feeder configuration problem for a chip shooter machine and investigate the case of one single machine and one board type case with the objective of minimizing the assembly (cycle) time per board (Chyu and Chang, 2008). The performance of the proposed algorithm, including the effect of feeder duplications, is presented and analyzed. They note that overuse of the feeder duplication will negatively affect the minimization of the assembly time per board. In the study, the constraint of multiple turret speeds is added to the problem. Even though they talk about different speed settings of board carrier table, they say that in practice, one widely used feeder arrangement policy is to assign the first few slots to the component types with a large population, that is, placing lighter components before heavier ones.

The formulation of objective function is based on placing one board and so takes special care about first six and last six turret rotations. But this formulation becomes wrong when continuous production is considered.

The solution procedure proposed in this paper has the following characteristics. First, component types using similar turret speed setting are clustered into a group. Suppose that there are K groups in total, with group k using the k^{th} setting. The speed for the mechanisms decreases as k increases, which implies that larger and heavier components are placed with little priority. Second, the algorithms with and without feeder duplications are both considered, and experiments are then conducted to observe the difference in their performances. Third, the decoding procedure of the GA-based algorithm involves a local search, and the mutation operator serves to move the current individual to another in a farther away neighborhood, and this new individual is then refined by the same local search method.

A feeder configuration list (FCL) contains a combination of sub-lists $\{FCL_k, k=1,..., K\}$. The turret rotation speed is set at a decreasing rate for k. The decoding procedure of an FCL consists of two steps: (i) constructing an initial placement sequence, and (ii) improving the initial placement sequence by 2-opt local search. For a FAL_k with no free movement, the NNH is applied to find the corresponding placement sequence following a slot-by-slot basis (it is important to note that the authors use the term nearest neighbor search, NNS instead of NNH). They call the NNS in this case 1-NNS, since the NNS is applied to one component type at a time.

For a FCL_k with a one-slot free movement, the 2-NNS method is applied to find the corresponding placement sequence.

The GA is applied on feeder configurations. In this proposed genetic algorithm, the order-based crossover is applied to each of these FCL_ks, which are then combined into an offspring. The tournament rule is used to select a pair of parents for a crossover operation. Likewise, the mutation operator uses cyclic replacement on each FCL_k with a probability of one-half.

They use two sample data sets. The first one contains 40 types with a total of 175 components, and the second sample has 50 types with 200 components. The comparison criterion is the calculated lower bound and they obtained assembly times which are about 8% far away from lower bound on the average. It is stated that the proposed method can be easily adjusted to apply to more recent machine models, such as CP 642 and CP 742, and to an assembly line consisting of multiple machines.

Iterative approach

Bard et al. addressed the placement sequencing, the feeder configuration and the component retrieval problems for a chip shooter machine (Bard et al., 1994). Their approach is an iterative two step approach. Problems are solved separately such that initially, a placement sequence was generated with a weighted nearest-neighbor TSP heuristic, while the remaining two problems were then formulated as a quadratic integer program and solved with a Lagrangian relaxation scheme. In the final step, the feeder configuration was used to update the placement sequence and the process was repeated. The methodology was simulated for a set of boards obtained from Texas Instruments and theoretically compared with a heuristic in use at the time.

Moyer and Gupta proposed the Acyclic Assembly Time (AAT) algorithm (Moyer and Gupta, 1996b) to optimize the placement sequencing and the feeder configuration problems simultaneously, as an extension to their previous study (Moyer and Gupta, 1996a). Actually this is a one step iterative algorithm that creates firstly a set of placement sequences and then feeder configurations.

The algorithm starts with generating an initial placement sequence using the nearest neighbor algorithm and then the placement sequence is improved using pair-wise exchanges. A fixed number of the best placement sequences are stored. After obtaining the placement sequences, an initial feeder configuration is constructed, combined by each of the placement sequences and improved using pair-wise exchanges. The placement sequence and feeder configuration arrangement that gives the best assembly time is chosen as the ultimate solution. They argued that their approach (Moyer and Gupta, 1996b) is superior to (Leu et al., 1993) and (De Souza and Lijun, 1994), on average.

Sohn and Park formulated the placement sequencing and feeder configuration problem as a nonlinear integer programming model for a chip shooter machine (Sohn and Park, 1996). They devised a heuristic algorithm to solve the problem with one head and used it to solve the multihead problem. The heuristic solves the problems sequentially by first assigning feeders to slots based on a frequency of use, and then determining a placement sequence by also considering feeder configuration. Computational experiences on some real world and randomly generated problems show that the algorithm for the one head problem also solves the problem with a multihead as well.

Yeo et al. employed a rule-based approach to generate the placement sequence first and then the feeder configuration for assembling a PCB on a chip shooter machine (Yeo et al., 1996). The following information about the machine is used. Travelling distance between placement operations should be at most 20 mm (in Chebyshev metric) and the free time of the feeder movement should be at most one slot. The developed approach was based on the one-pitch incremental feeder heuristic and the nearest neighbor heuristic. Four rules are developed for reaching the goal of reducing the board carrier and feeder carriage movements in order to maximize the machine throughput. Computational results showed that the proposed approach outperformed the Fuji machine optimization software in terms of the PCB table movement and the actual machine cycle time. About 25% improvement is obtained in terms of total production.

Ng deals with the chip shooter machine but its name is not spelled within the study (Ng, 1998). He proposes a heuristic solution technique of low computational complexity to find a better assembly plan comprising of the placement sequence of electronic components and the feeder configuration. The developed algorithm combines the physical constraints of the placement machine and a `grouping' concept that takes advantage of the natural structure of a PCB.

The algorithm firstly finds the placement order of reels and then the placement sequence. Lastly, placement sequence is improved by exchanges. Also, a procedure is proposed for determining the types of components on the multiple reels when number of types of components is less than number of feeder reels. He uses real data obtained from a PCB production plant in which the optimization was done solely by human experience. Six data sets were used and all of them are improved by 15% on the average.

In (Ng, 2000), Ng makes an extension to his previous study (Ng, 1998). Even though he does not pronounce chip shooter term, the defined machine is exactly a chip shooter with 10 placement heads and 70 feeder reels. He proposes a solution technique based on the clustering approach which addresses four problems simultaneously; (1) determination of placement sequence, (2) allocation of component types to feeder reels; (3) feeder configuration; and (4) component retrieval so as to minimize the total assembly time of a PCB.

He says that in the previous study it was assumed that the number of types of components (n) is less than number of feeder slots (f), i.e $n \leq f$. But this study makes an assumption of $n > f$ (that is, in this case it is needed to initialize the PCB machine a number of times). The solution is reached using clustering approach. When $n > f$, PCB can be produced by inserting components in a number of sub-assemblies of the PCB production process.

A clustering-based heuristic algorithm is proposed. To evaluate the performance of the proposed algorithm, a convenient lower bound for each sub-assembly time is derived and used. Three sets of real data is obtained from the plant. He compares the performance of their heuristic algorithm with the existing solutions used by the plant. The results show that their algorithm is quite effective. On average of these three sets of data, the percentage improvement is about 13.57%.

In (Ellis et al., 2001) and (Vittes, 1999), a new solution approach to determine the component placement sequence and feeder arrangement for a chip shooter machine is developed. Actual PCB data is used to demonstrate the performance of the proposed solution approach. After giving a literature survey for combined feeder arrangement and placement sequencing problems, it is stated that current literature assumes all components are mounted using the same turret rotation time, feeders occupy only one feeder slot and board carrier speed and feeder carriage speed are constant. Many of these restrictions are relaxed in their research. For example, a heavy component may reduce the speed of the board carrier as well as the speed of the turret. Besides, the placement time for the first component is assumed to be the fixed pick and place time and for the last $H/2$ components the feeder carriage movement time is assumed to be zero since there are no more components to pick up. But this assumption is erroneous when continuous production is considered.

The heuristic solution constructs an initial solution for the problem using a set of rules that are based on the functional and operational characteristics of the placement machine. These rules stay very specific to the specified machine and it is hard to apply to another chip shooter machine. The initial solution is then improved using the two-opt improvement algorithm.

The proposed heuristic uses arbitrary insertion point algorithm (AIP) and before explaining the heuristic, it is beneficial to give the definition of AIP. The algorithm consists of initially selecting two components to form a partial placement sequence. The two components chosen are the component with the lowest feeder number and largest *x* coordinate and the component with the highest feeder number and the smallest *x* coordinate with the same board carrier speed. Then a new component is added to the component placement sequence at each stage, with a total of N-2 stages. Thus a partial placement sequence always is maintained, and the placement sequence increases by one component at each stage. At each stage, the unassigned component with the smallest *x* coordinate is selected. The component selected is added to the component placement sequence in the place where it increases the total movement time the least. This procedure is repeated until all the components are assigned to the placement sequence.

Initial solution consists of five steps. In the first step, the components having the same board carrier speed and turret rotation speed are grouped together. Secondly, for each group an initial component placement sequence is created using the NNH (the component with smallest *x* coordinate is selected as initial location). Thirdly, a component type flow matrix is constructed in the way that if component type *i* is placed after component type *j* then the flow between component types *i* and *j* and component types *j* and *i* is increased by one. In the fourth step, an initial feeder configuration is created by using a greedy algorithm and flow matrix. The greedy algorithm develops a feeder sequence by placing the component types with the highest flow close together in the feeder carriage. The configuration starts at the left of the feeder carriage with the group with fastest board carrier speed and slowest turret rotation speed and proceed to the right of the feeder carriage while taking into account the feeder space required by each component type. Finally, the placement sequence is reformed by using AIP algorithm or the NNH. The component with the lowest feeder number and smallest *x* coordinate is selected as the starting component for the placement sequence. The placement time between two consecutive component placements is measured by a function that considers movements of all three moving parts and penalizes any change in board carrier speed.

Placing components with a faster board carrier speed first helps to avoid reducing the board carrier speed at the beginning of the component placement sequence. One of the restrictions they impose is that once the board carrier speed is shifted to a lower speed value, it stays at that low speed value and does not increase while populating the PCB. Thus, placing components with slower table speeds first would increase the PCB table movement time. Moreover, placing components with low turret rotation speed first, within each table speed setting is often advantageous. If the lower rotation speed components are placed first, then components with high turret rotation speed are not required to be placed at the turret rotation speed associated with components with low turret rotation speed. This is because the low turret rotation speed components will no longer be in the turret placement heads when the time to place the components with a high turret rotation speed arrives. Remember that the turret rotation time associated with the i^{th} component placement is dictated by the component with the lowest turret rotation rate loaded in the turret (Vittes, 1999).

The improvement phase consists of separate improvements of placement sequence and feeder configuration, iteratively. Both improvements are made using 2-opt algorithm. The iteration stops when no improvement is obtained in one of the problems.

The component data for four PCBs (labeled PCB 1, 2, 3, and 4) were obtained from Ericsson's facility. The number of components placed ranged from 129 to 1661 and the number of component types ranged from 19 to 69. PCB 4 is not an actual PCB, but instead a combination of several PCBs with their x and y coordinates combined. For all PCBs in the case study, the proposed solution approach yielded lower placement times than the commercial software. However, although the final solution was close to the lower-bound, the time to generate improved solutions was high, and become worse as the problem size is increased.

Kumar and Luo present a formulation and solution of the assembly time optimization problem for chip shooter machine (Kumar and Luo, 2003). Chip shooter term is not spelled within the study, rather ‘chip placement machine’ is used. A near-optimal solution is obtained by relaxing the problem to an instance of TSP. This is achieved by using a realistic assumption that feeder carriage movement is considerably slower compared to the movement of the PCB table and rotation of the turret, they transform this nonlinear problem to an instance of “generalized TSP”, where not only the overall travel time is a function of the travel sequence (as in standard TSP), but even the distances between any pair of nodes depend on the travel sequence (in contrast, in standard TSP these remain constant). It should be pointed out that the problem that the authors refer as a “generalized TSP” is actually the Sequence Dependent TSP (SDTSP) and it is explicitly introduced to the literature in this thesis. The structure of the problem allows us to further approximate this problem into an instance of standard TSP under the assumption given above. Under these assumptions, in order to minimize the overall travel time for such motions one must minimize the number of feeder carriage movements. The authors propose a method that places all components of the same type consecutively so that feeder carriage only moves each time a new type of components is started to place. On the other hand, the feeder configuration problem vanishes by the above simplifying assumption and is not considered in the algorithm. Therefore, the authors easily conclude that feeder configuration can be made arbitrarily because an optimal tour visiting all components of the same type consecutively will always adjust itself with respect to the given feeder configuration such that whenever a switch from one component type to another is made it is between component types occupying adjacent slots. It is also advised to place heavier components toward the end of assembly process by assigning heavier component types to final slot locations since once the heavier components are placed on the PCB, the board carrier can only be moved slowly.

The results of the proposed algorithm are compared against those currently used in industries via three actual PCB data; it demonstrated a saving in assembly time of more than 25% in all cases considered. These three PCB (named as A, B and C) contain 129, 496 and 867 components and 19, 58 and 95 component types, respectively.

The generalized TSP pronounced by the authors cannot be figured out explicitly in the study. Besides by the simplifying assumption (given above), the problem is defined as a standard TSP. On the other hand, they take into account another realistic assumption that a heavier component on the board imposes a limit on the acceleration of the platform since otherwise the components will move. To solve this problem they offer to assign the heavier component types to the final slot locations so that they are placed toward the end of the assembly operation.

Solving PS, FC and CR problems separately

Even though it has a little use in the chip shooter machines, some researchers worked on optimizing one of PS, FC or CR problems by assuming the other two are solved and given as an input (rather than as a constraint) to the problem.

Horak and Francis address the feeder configuration problem of PANASONIC Mk 1 chip shooter machine (Horak and Francis, 1995). They develop an approach which uses placement sequence as input.

Moyer and Gupta formulated the feeder configuration problem as a QAP for a given placement sequence for chip shooter machines (Moyer and Gupta, 1996a). Thus, they aim to minimize the feeder travelling distance. They proposed two heuristic approaches. First one is a construction (Feeder Slot Allocation, FSA) heuristic which assigns feeders to slots based on the switching between component types according to the given placement sequence. The second heuristic is an improvement heuristic based on pair-wise exchanges and is presented for comparison with the FSA heuristic. Reasonably good solutions are obtained by the FSA heuristic in a short running time. On the other hand, the improvement heuristic yields equal or better solutions than the FSA heuristic, but has a longer running time. Besides, they obtained better feeder configuration, compared to (Leu et al., 1993) in terms of feeder travelling distance.

De Souza and Lijun developed a knowledge-based component placement system (CPS) incorporated with TSP algorithms to address the placement sequencing for the chip shooter machine (De Souza and Lijun, 1995). The algorithm first groups the components by type, then by a quantity threshold and finally by the device size. They pointed out that the CPS is more practical and superior when compared with the software supplied by the machine vendor by about 24%.

Moyer and Gupta address the component sequencing problem for a chip shooter machine (Moyer and Gupta, 1997). By their definition, this machine configuration consists of a moveable X-Y positioning table and a tape-and-reel sliding feeder rack. In this machine type, the distance metric used for the travelling distance between components is the Euclidean metric instead of the Chebyshev metric. They define the placement sequencing problem as a TSP and focus on solving it by ignoring the feeder configuration problem. The basis for this ignorance is the assumption that the PCB table movement time is generally more time consuming than the feeder carriage movement time. Thus, minimizing the travelling distance between the components on the PCB has a higher priority in their research.

The board sequencing heuristic (BSH) was developed to solve the problem. The idea of the BSH was to rearrange the placement order by swapping a pair of components (pair-wise exchange) in the current tour in order to obtain a better solution. For generating an initial placement sequence three alternative methods are described. These are, generating a placement sequence based on a random selection, based on an increasing component type identifier, and based on a sorting scheme of X coordinates, and then Y coordinates of the components. The heuristic approach is tested against a previously published subproblem as well as a real-life working board configuration containing 266 components. The BSH improved the initial random solution by 81%, the initial solution based on the component type identifier by 52%, and the initial solution based on sorting the components based on the X-Y coordinates by 64%.

Ho and Ji present a HGA to determine the component sequencing problem for a chip shooter machine with the objective of minimizing the total traveling distance of the board carrier (Ho and Ji, 2005). The algorithm is the same as the one in their former study (Ho and Ji, 2003) as we summarized above. It is stated that, none of the researchers have studied the component sequencing problem for the chip shooter machine using a HGA. Besides, the performance of the HGA is compared with the BSH developed by (Moyer and Gupta, 1997) and outperformed it.

Crama et al. deal merely with component retrieval problem on a chip shooter machine Fuji CPII (Crama et al., 1996). Component retrieval problem can be defined as follows: for a given placement sequence of components on the board, and for a given assignment of component types to (possibly multiple) feeder slots of the placement machine, decide from which feeder slot each component should be retrieved.

They reformulated the component retrieval problem as a longest path problem in a PERT/CPM network with design aspects; finding the minimal makespan of the assembly process thus amounts to identifying a design for which the longest path in the induced network is shortest. Alternatively, they viewed the problem as a shortest path problem with (path induced) side-constraints. They also showed that forward dynamic programming cannot be used to obtain optimal solutions. They propose an algorithm to solve the problem which has a complexity of *O(ve)* with *v* nodes *e* edges.

Grunow et al. mathematically models the operations of chip shooter machines and develop a simulation system for practical use in the electronics assembly which incorporates various types of assembly machines (Grunow et al., 2000). The author emphasizes that almost all of the studies analyzing chip shooters neglect some fundamental technological constraints:

- The dependence of the turret rotation speed on the component type is not considered.
- The varying width of the component feeders is not accounted for.

They also developed a simulation system called EASE with the properties simulating the chip shooter machines.

II.3.5.2. Studies on chip shooters in other production environments

Dikos et al. did not study PS problem, the assumption is made that an optimal component placement order for a card design will be specified by experienced operations engineers using software from the equipment vendor (Dikos et al., 1997). Thus, they concentrate on the optimization of the FC problem. They also consider solving feeder arrangement for multisetup strategy in which one assembly setup is utilized for manufacturing a variety of circuit cards. Various crossover operators and selection operators are tested within the developed GA. They argue that the results look promising.

In (Ohno et al., 1999), the authors deal with assembling multi-type PCBs with 18 placement heads and 100 feeder slots. They argue that the multi-type PCB assembly (MPCBA) optimization problem is proposed for the first time. It is a combination of three subproblems: a PS problem, a FC problem, and an optimal assembly mode problem (OAMP). They define assembly mode and OAMP as follows: An assembly mode means a partition of all types of PCBs into subsets. The OAMP is concerned with the optimal partition of the PCB types into subsets such that all the components required for each subset fit on the reel carrier. All the lots of each PCB type in a subset are assembled, one following another, with no feeder setup time between the different types since all their components are already on the feeder carriage. Setup is needed only in the changeover between subsets. An optimal assembly mode minimizes the sum of assembly times and setup times of all the demanded PCBs.

The PS problem and the FC problem actually consist of one FC, which positions the components on the reel for the PCBs in a subset, and K_j PSs, each corresponding to inserting the components in PCB type k, k=1,…,K_j and K_j is the number of PCB types in group j.

The PS problem for each type of PCB is formulated as the TSP for a fixed feeder configuration and the FC problem is formulated as the assignment problem for which the assignment cost is the sum of the weighted tour costs of the TSPs for a subset of PCB types. The PSs and FC is solved by the composite algorithm recursively by alternately solving the PSs by a heuristic based on the 2-opt procedure, and the FC by a heuristic based on the evolution strategy procedure.

On the other hand, OAMP can be formulated as the set partitioning problem (SPP) with TSP type constraints. A heuristic is devised that is based on the composite algorithm given above. The developed techniques are compared on a real life problem with current assembly techniques and showed better performance. The real life problem contained 18 PCB types and they divided them into four main types of PCBs.

Chen and Chyu analyze the multiple PCB types one machine case (M-1) and solution methodology for three subproblems, PCB grouping, group feeder assignment and placement sequence optimization is developed (Chen and Chyu, 2002). The machine used in the study is a chip shooter machine. For grouping PCBs, three methods are proposed that incorporate three criteria, i.e. free slots remaining, mutual commonality components and time of placement penalty to compute similarity measures between PCBs. In the study, a new weighting similarity measure is defined. In order to compare these three group management strategies and test the efficiency of the new similarity measure, test instances are generated (12 simulated boards) and solved with each of the procedures. To solve FC and PS problems for a family of PCBs, an efficient procedure based upon an ant colony optimization (ACO) algorithm is developed. It is compared with sequential method (feeder assignment and placement sequence are solved recursively by alternately improving the feeder assignment by a heuristic based on the 2-opt procedure, and improving the placement sequence by a heuristic based on the simulated annealing), and genetic algorithm using six hypothetical PCBs. ACO approach outperformed both comparative approaches. Also a PCB data from (Leu et al., 1993) is used for evaluating the performance of ACO algorithm.

Ammons et al. list the basic four problems in process planning of PCB manufacturing; selecting setup management strategy, component allocation, FC and PS (Ammons et al., 1993). For feeder configuration and placement sequencing problems, their solution approaches rely on previously developed heuristics that solve these two problems. The concurrent machine depicted in the study is a chip shooter machine. In their case studies, data from several different manufacturing operations, with various card types and assembly machines is used. Both the unique setup strategy and the family setup strategies are considered in this case study. Two scenarios have been analyzed for the unique setup strategy: single machine/single card setup and multiple machine/single card setup. Two scenarios have also been analyzed for the family setup strategy: single machine two card setup and the multiple machine/two card setup.

Crama et al. deal with problems in M-machines M-board type environment (Crama et al., 1997). After identifying the problems, a two phased solution approach is presented. The solution procedure is tested on real-life instances and it is shown that the current solutions are substantially improved. The analyzed machine is a chip shooter (Fuji CP-IV). They assume that the problem of “partitioning of the set of board types into families which are to be assigned to different lines of placement machines, and obtaining a sequence of the board types within each family, indicating in which order these board types will be produced", has been solved; thus, they deal with a set of different types of PCBs (a family) that has to be produced by a line of several placement machines.

In their paper, the problems they focus are defined as below. By focusing these problems they deal with planning problems that arise when a given family of board types is assembled by a single line of placement machines.

(1) for each board type, a partition of the set of component locations on this board, that is a decision concerning which locations are going to be served by which machine,

(2) for each machine, a feeder rack assignment, that is an assignment of feeders to positions in the feeder rack,

(3) for each pair consisting of a machine and a board type, a component placement sequence, that is an order in which components are placed at the locations on this board that are served by this machine, and

(4) for each pair consisting of a machine and a board type, a component retrieval plan, that is, for each component on the board, a rule indicating from which feeder this component should be retrieved.

One of the observations that is caught is about the free movements. The movements of the feeder carriage and the board carrier, taking place during a carousel rotation, are referred to as free movement. They observe that, free movements of the feeder carriage correspond to one position on the carriage, that is, repositioning the feeder carriage by zero or one slot is free. Concerning the board carrier, free movement corresponds to approximately two cm on the table containing the PCB. Since the average PCB is approximately 20 × 30 cm, and the feeder carriage contains mostly about 100 slots, long board carrier movements are less time consuming than long feeder carriage movements. Hence they try to restrict the planning procedure to considering solutions in which all feeder carriage movements are short, expecting that given this short feeder carriage movements they can find a PS in which the board carrier movements are not too long either.

For solving the above mentioned problems, they propose a procedure. This procedure is divided into two phases: Phase 1 determines a feeder configuration for each machine, and Phase 2 produces, for each pair consisting of a machine and a board type, a component placement sequence and a component retrieval plan, given the feeder configuration of Phase 1. Hence, it is easily concluded that this study solves the FC and PS problems separately, not even iteratively. At this point the following question arises: how does the feeder configuration computed during Phase 1 is evaluated without computing a placement sequence and retrieval plan, that is, without solving Phase 2? They answer the question as follows: They deal with this issue by computing an estimate of the makespan of each board type on each machine given the FC and a corresponding partition of the components of each board type. These estimates are then used as an indication of the quality of the feeder configuration found during the execution of the algorithm. Within the solution approach, the component retrieval problem is modeled straightforwardly as a shortest path problem and solved thereafter.

They test the proposed planning procedure on two datasets. Dataset 1 corresponds to a family consisting of nine board types assembled by a line of three CP-IV machines. Dataset 2 corresponds to a family consisting of seven board types assembled by a line of two machines, a CP-IV and a CP-III machine (They assume that the Fuji CP-III operates in an identical fashion as the Fuji CP-IV, but at a different speed). These datasets are real life data made available to them by Philips.

Another important note about the paper is that they remind the reader about the change of turret time speed depending on the carried components. However, while modeling the machine they neglect this property for simplicity and argue that this property of the machine will have only marginal effects to the assembly times, which we don't agree.

Klomp et al. made an extension to the algorithm presented in (Crama et al. 1997) considering new constraints and situations (Klomp et al., 2000). Their feeder rack assignment problem is defined as : Given a line of chip shooter machines and a given family of boards find a heuristic to answer the question 'where (in what slots) to attach the feeders alongside the feeder racks of the placement machines', optimally. Thus, they want to construct a single feeder configuration for all machines in the line that allows good component placement sequences for all board types in the family. Ahmadi et.al. also addresses the same question under a different name 'reel positioning problem' (Ahmadi et.al., 1995). In the single machine, single board case, the feeder rack assignment problem turns out to be feeder configuration problem. Klomp et al. treated the problem as finding a shortest Hamiltonian path (Klomp et al., 2000). An insertion heuristic and a local search heuristic were employed to solve the problem. They also considered feeder duplication which is the presence of more than one feeder of some type in the assembly line. The computational results are carried out for placement lines consisting of two or three Fuji CP-IV/3 placement machines. They used two real data sets, one corresponding to a family of two board types (data set 1) and one corresponding to a family of 20 board types (data set 2); data set 1 concerns 873 components of 115 types and data set 2 concerns more than 8200 components of 123 types. Their results demonstrated that the gap between the solution found and the lower-bound was about 20% in the three machine case. The performance of the algorithm is compared to Fuji software in use at the time and in case of no feeder duplication their method yields an improvement of about 14%. The results include an interesting conclusion that there is no advantage in assembly time reduction when feeder duplication is used.

II.3.5.3. TSP and QAP arising in other placement machines and environments

Walas and Askin developed iterative approach for solving the TSP and QAP concurrently for numerical control punch presses (Walas and Askin, 1984). By generating a feasible solution and then alternately improving the subproblems, good solutions are obtained in reasonable time.

Gavish and Seidmann address a concurrent machine with a movable feeder carriage (in one dimension), movable board carrier (in two dimensions) and a pick and place head that picks components from a fixed location and places them to fixed location (Gavish and Seidmann, 1987). The placement sequencing problem is formulated as a TSP and the feeder configuration is formulated as a QAP and they are solved iteratively.

Leipala and Nevalainen addressed the placement sequencing and feeder configuration problems of the Panasert RH machine, which is a turret type machine with two placement heads, a moving feeder magazine and a moving board carrier (Leipala and Nevalainen, 1989). The placement sequencing problem for a fixed feeder configuration is formulated as a three-dimensional asymmetric traveling salesman problem. On the other hand, the feeder configuration problem for a fixed insertion sequence is formulated as a quadratic assignment problem. The two problems were solved sequentially by successive fixing of the placement sequence and the feeder configuration. They suggest a pairwise exchange heuristic for the feeder configuration and a modified furthest insertion heuristic for the placement sequencing problem. Additionally, they talk about the initial feeder assignment and suggest several alternative techniques.

Crama et al. deal with optimizing the operations of CSM-60 placement machine together with the assembly line in which it is placed (Crama et al., 1990). They define six problems for this assembly line and model them by well known NP-Complete problems. For solving these problems best methods are proposed. Especially, feeder configuration problem is modeled as QAP and determination of placement sequence problem is modeled as TSP. The interdependency between them is emphasized and they are solved iteratively. Applied technique for TSP is nearest neighbor and farthest insertion methods. For QAP local improvement method based on pair wise exchange of feeders is applied.

In (Ho et al., 2007), the analyzed machine is collect and place machine (very different from chip shooter) but the problems are formulated as TSP and QAP. The HGA that they develop in previous studies is applied to this type of machine. The solution is represented by a chromosome with the three link path representation. Link 1 represents the sequence of component placement, link 2 denotes the assignment of component types to feeders next to first head (Head A), whereas link 3 denotes the assignment of component types to feeders next to second head (Head B). Within GA, after crossover and mutation, improvement heuristics are applied to improve the solution. These are nearest neighbor heuristic and iterated swap procedure.

In (Duman and Or, 2007), a search is made among those metaheuristics that have recently found widespread application in order to identify a heuristic procedure that performs well with the QAP in the PCB assembly context. The focus is on machines where the feeder configuration problem can be formulated as a QAP. In these machines placement head is fixed, the board carrier is movable in two dimensions and the feeder carrier is linear and movable in one dimension. One such machine is the Panasonic variable center insertion machine.

They compared four classes of heuristic approaches and their variants with different parameters reaching a total of 44 procedures. Namely, these main heuristic approaches are exchange procedures, taboo search, simulated annealing and guided evolutionary simulated annealing.

Performance of these various QAP solution procedures is investigated and compared using two classes of test problem. On problems arising from real PCB assembly environment SIMANH14, SIMANH8, SIMANH13, SIMANH12, GESAH36 showed best performance, while on problems with general characteristics k-OPT and taboo search showed best performance.

II.4. RELATED COMBINATORIAL OPTIMIZATION PROBLEMS AND SOLUTION APPROACHES

Combinatorial Optimization deals with a special type of mathematical programming problems with the property that the set of feasible solutions corresponds to a finite set. In its most general form such a problem can be stated as follows (Jünger et al., 1995).

Given a finite set S (set of feasible solutions) and a function $f{:}S \rightarrow R$, find an element $s^* \in S$ with $f(s^*) = \max\{f(s) \mid s \in S\}$.

Such a general optimization problem is of no use unless we have a reasonable characterization of the set of feasible solutions S and an algorithmic way for evaluating the objective function value for each $s \in S$.

A large number of real world planning problems turn out to be combinatorial optimization problems all of which are easy to state but have a finite very large number of feasible solutions. While some of them such as Shortest Path problem and Minimum Spanning Tree problem have polynomial time algorithms, most of them such as Traveling Salesman problem and Quadratic Assignment problem have no polynomial method for their solutions.

Combinatorial optimization problems arising in the context of PCB assembly systems aims either minimization of a value (such as setup time, placement time etc.) or maximization of another value (such as throughput, workload balance etc.) under certain constraints (such as available feeders, machine speed etc.).

In this section, we will give a classification of the combinatorial optimization problems using some mathematical models and then the general solution techniques developed for them.

Combinatorial optimization problems are solved using mathematical programming models. A mathematical programming model is a mathematical representation of the actual situation that may be used to make better decisions. Linear programming model, integer linear programming model and nonlinear programming model are basic models for solving various combinatorial optimization problems, especially the problems that this thesis focuses on.

A model is defined as a linear program when the objective function and the constraints are linear expressions and the decision variables are continuous. The transportation problem is one of the LP examples. The transportation problem was first formulated by Hitchcock in 1941; he also gave a computational procedure, much akin to the general simplex method, for solving the problem (Hitchcock, 1941). When compared with integer linear programming and nonlinear programming problems, linear programming problems are vey well studied and can be solved even on very large scale. Other then the well-known simplex algorithm introduced by Dantzig, interior point methods are the techniques developed for LP (Dantzig, 1951, Lustig et al.,1992).

Unfortunately, only a minority of practical problems can be modeled as pure linear programming problems. In many cases, some or all of the variables have to take integer values. Hence integer linear programming (or integer programming) (IP) is widely adopted as a method of modeling. It should be clear that LP is a subset of IP and we refer to LP as the linear programming relaxation of IP. Generally, there are three types of IP. First, an IP is called a pure integer linear programming if all variables are required to be integers. Second, an IP is called a mixed integer linear programming if only some of the variables are required to be integers. Third, an IP is called a binary integer linear programming if all the variables must be either 0 or 1.

IP has important practical applications. Though, it was stated that computational experience with IP has been less than satisfactory (Taha, 2003). An IP computer code that can solve IP problems consistently does not exist.

II.4.1. The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is one of the most widely studied IP problems. Adding new constraints to the problem yields different generalizations to the problem and each new generalization forms the basis of a new research area. Mostly known generalizations are Asymmetric TSP, Vehicle Routing Problem (VRP) and its variants. TSP is observed in many research areas. One can easily argue that there is uncountable number of studies towards solving it since it is introduced to the literature, ranging back to at least the late 1940's. The TSP is the focus of interests for many research disciplines (mostly computer scientists, mathematicians and industrial engineers) because, even after about half a century of research, the problem has not been completely solved. TSP or its variants can be applied to solve many realistic problems within our daily lives.

It can be easily stated as follows: A salesman wants to visit *n* distinct cities and then returns home. He wants to determine the sequence of the travel so that the overall travelling distance is minimized while visiting each city not more than once. Conceptually it seems simple, but obtaining an optimal solution requires a search in set with *n!* feasible solutions. TSP can be represented on a graph *G=(V,E)*, where *V* is the set of vertices (or cities) and *E* is the set of edges (or links between the cities). Each edge $e \in E$ has an associated cost (or length) c_e. We define the incidence vector *x* as

$$x_e = \begin{cases} 1, & \text{if edge e is used} \\ 0, & \text{otherwise} \end{cases}$$

Notice that for a tour, at each vertex the sum of the edge variables must be two; this is called a degree constraint. This leads to the relaxation of the traveling salesman problem:

$$\min \sum c_e x_e \tag{II.7}$$

$$\text{s.t.} \sum_{e \in \delta(v)} x_e = 2 \quad \text{for all vertices v} \tag{II.8}$$

Here, $\delta(v)$ denotes the set of all edges incident to vertex *v*. All tours are feasible in this formulation, but it also allows infeasible solutions corresponding to subtours, consisting of several unconnected loops. To force the solution to be a tour, it is necessary to include subtour elimination constraints of the form $\sum_{e\in\gamma(u)} x_e \geq 2$ for every subset $U \subseteq V$ with cardinality $2 \leq |U| \leq V/2$, where $\gamma(U)$ denotes the set of edges with exactly one endpoint in *U*. Any feasible solution to the relaxation given above which also satisfies the subtour elimination constraints must be incidence vector of a tour. One can see that the number of subtour elimination constraints is exponential in the number of cities (Mitchell, 2002).

Another formulation of TSP is done by using variables x_{ij} representing the travel of the salesman from city *i* to city *j*, and c_{ij} representing the cost of travel.

$$\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij} x_{ij} \tag{II.9}$$

$$\text{s.t. } \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,n; i \neq j \tag{II.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,n; i \neq j \tag{II.11}$$

$$u_i - u_j + n x_{ij} \leq n-1 \qquad i, j = 2,3,\ldots,n; i \neq j \tag{II.12}$$

$$x_{ij} \in \{0,1\}, \qquad i = 1,\ldots,n, \qquad j = 1,\ldots,n, \tag{II.13}$$

$$u_i \in Z^+ \qquad (Z^+ = \text{set of positive integers}) \qquad i = 1,\ldots,n, \tag{II.14}$$

In this formulation, first constraint ensures that the salesman arrives once at each city and second constraint ensures that the salesman leaves each city once. Avoidance of subtours is done by third constraint (Winston and Venkataramanan, 2003).

In general TSP, there is no restriction on the cost function. In metric TSP, all edge cost are symmetric and fulfill the triangle inequality: $c_{ij} \leq c_{ik} + c_{kj}$. In Euclidean TSP, vertices correspond to points in a *d*-dimensional space, and the cost function is the Euclidean distance. The Euclidean distance between two points $x=(x_1, x_2,\ldots,x_d)$ and $y=(y_1, y_2, \ldots, y_d)$ is

$$\sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{II.15}$$

For a complete survey and summary about TSP, we direct the reader to (Gutin and Punnen, 2002).

II.4.2. The Quadratic Assignment Problem (QAP)

For LP and IP, the objective is to minimize or maximize a linear function with linear constraints. Even though LP and IP problems are very common and cover a wide range of problems, there are cases where the function or one of the constraints is not linear in real-life. If a problem contains a nonlinear term, then it is classified as nonlinear programming (NLP) problem.

A well-known example of NLP problems is the Quadratic Assignment Problem (QAP). The simplest interpretation of the QAP uses assignment of offices to people as follows (Hanan and Kurtzberg, 1972). In the QAP, we are given a cost matrix $C = [c_{ij}]$, where c_{ij} is the measure of the affinity between person i and person j. We are also given n possible offices to which we can assign the people. Finally, we are given a so-called distance matrix $D = [d_{kl}]$; where d_{kl} represents the distance between office k and office l. Assume that person i is assigned to office $p(i)$, and that person j is assigned to office $p(j)$. Then the cost associated with this assignment is taken as $c_{ij}d_{p(i)p(j)}$. Thus, we see that the total cost of all office assignments will be the sum of each $c_{ij}d_{p(i)p(j)}$ over all i, j. The optimal assignment will be that one in which the total cost is minimal. "If the affinity represents the amount of 'face-to-face communication', then the assignment which we desire is the one which minimizes the total amount of walking distance for the people" (Hanan and Kurtzberg, 1972).

A formulation of QAP can be given as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ij} d_{kl} x_{ik} x_{jl} \qquad \text{(II.16)}$$

s.t.

$$\sum_{i=1}^{n} x_{ik} = 1, \qquad k = 1,\dots,n \qquad \text{(II.17)}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \qquad i = 1,\dots,n \qquad \text{(II.18)}$$

$$x_{ik} \in \{0,1\}, \qquad i,k = 1,\dots,n \qquad \text{(II.19)}$$

This formulation uses the binary variable x_{ik} which states the assignment of person i to office k.

Mostly, QAP is defined in conjunction with Linear Assignment Problem (LAP) in order to demonstrate the nonlinearity and hardness of the problem. A common definition of the LAP involves the assignment of n people to n jobs. For each job assignment, there is a related cost, c_{ij}, of assigning person i to job j. The objective is to assign each person to one and only one job in such a manner that minimizes the sum of each assignment cost, i.e., the total cost (Hanan and Kurtzberg, 1972). Mathematically,

$$\min \sum_{i=1}^{n} c_{i\pi(i)} \tag{II.20}$$

over all permutations $\pi \in S_n$, where S_n is the set of permutations of 1,2,…,n and $j = \pi(i)$ is the job assignment of person i.

In both QAP and LAP there are $n!$ permutations from which to choose the optimal assignment. However, it should be noted that there is a key difference between these two problems which makes the QAP considerably more difficult to solve. Unlike the LAP in which the assignment of job j to person i was made independently of the assignments of the other employees, with the QAP the assignments are not independent. That is, when considering an assignment of person i to office k, one must consider the assignments of all other people who have some nonzero affinity for person i.

II.4.3. The Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is one of the most challenging combinatorial optimization problems. In the most general terms, it consists of designing routes for a fleet of vehicles which are assigned to visit a given set of customers with the aim of minimizing total cost. It was introduced to the literature by (Dantzig, 1959) almost 50 years ago. Due to its complexity, it is an NP-Complete problem. The formulation of the problem can be stated as follows.

Suppose there is a uniform fleet of M vehicles, each with capacity C, to serve geographically dispersed customers, each with a deterministic demand that must be serviced by a single vehicle. Let V be the set of n demand nodes and a single depot, denoted as node 0. Let d_i be the demand at each node i. We consider the fully connected network, and denote the deterministic travel time between node i and node j by c_{ij}. We define integer variables x_{ij} which indicate whether a vehicle goes from node i to node j or not. In order to eliminate subtours, we include continuous variables u_i for every i in $V\setminus\{0\}$ that represent the flow in the vehicle after it visits customer i.

$$\min \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{II.21}$$

s.t.

$$\sum_{i\in V} x_{ij} = 1 \qquad j \in V\setminus\{0\} \tag{II.22}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad i \in V\setminus\{0\} \tag{II.23}$$

$$\sum_{i\in V} x_{i0} = M \tag{II.24}$$

$$\sum_{j\in V} x_{0j} = M \tag{II.25}$$

$$u_j - u_i + C(1-x_{ij}) \geq d_j \qquad i,j \in V\setminus\{0\}, \quad i \neq j, \quad d_i + d_j \leq C \tag{II.26}$$

$$d_i \leq u_i \leq C \qquad i \in V\setminus\{0\} \tag{II.27}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{II.28}$$

Constraints (II.22) and (II.23), force that a single vehicle goes into and out of every node; constraints (II.24) and (II.25) ensure that a total of K vehicles leave the depot and then return to it; constraints (II.26) and (II.27) are subtour elimination constraints imposing both the capacity and connectivity of the feasible routes.

Imposing several more constraints on the problem creates several variants of the VRP which can be listed as follows:

- Every vehicle has a limited capacity (Capacitated VRP - CVRP)
- Every customer has to be supplied within a certain time window (VRP with time windows - VRPTW)
- The vendor uses many depots to supply the customers (Multiple Depot VRP - MDVRP)

- Customers may return some goods to the depot (VRP with Pick-Up and Delivery - VRPPD)
- The customers may be served by different vehicles (Split Delivery VRP - SDVRP)
- Some values (like number of customers, theirs demands, service time or travel time) are random (Stochastic VRP - SVRP)
- The deliveries may be done in some days (Periodic VRP - PVRP)

To solve the VRP, several constructive and improvement based algorithms are proposed from the day it is introduced. A review of these can be found in (Laporte, 1992). Because of differing constraints, a general solution approach that manages to give good results for all VRP variants cannot be achieved. Hence each variant turns out to be a special topic for research. For CVRP, (Toth and Vigo, 2002) covers most important branch and bound based algorithms proposed for asymmetric and symmetric CVRP.

II.4.4. The Time Dependent TSP (TDTSP)

The Time Dependent TSP is a generalization of the TSP in which cost of travel between node *i* and node *j* depends not only on the respective locations involved, but also on their positions in the sequence that defines the tour (Picard and Queyranne, 1978). TDTSP is studied generally in the context of single machine scheduling algorithms and is not studied in the context of placement machines in the literature. Another similar problem assumes varying travel costs depending on the time zone of the day due to several causes of which mostly pronounced is the traffic jam of the considered city (Schneider, 2002; Malandraki and Daskin, 1992; Ichoua et al., 2003; Haghani and Jung, 2005; Albiach et al., 2008). This generalization of TSP is also referred as TDTSP in the literature, but this definition of TDTSP is out scope of this thesis.

In the literature, given its importance in applications, a number formulations has been proposed for the TDTSP (Picard and Queyranne, 1978, Wiel and Sahinidis, 1996, Fox et al., 1980). A summary of these present different formulations of the TDTSP are given by (Bigras et al., 2006). Of these formulations, we will adopt the one given by (Wiel and Sahinidis, 1996).

In an undirected graph *G(V,E)*, let *V* denote the set of nodes and *E* denote the set of edges. If there exists an edge between nodes *i* and *j*, then c_{ijt} is the cost of travel from node *i* to node *j* when it is made in time period *t*.

Two sets of binary variables are used in the formulation:

$$x_{it} = \begin{cases} 1, & \text{if node i is visited in time period t} \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijt} = \begin{cases} 1, & \text{if node j is visited in time period t after city i was visited in time period t - 1} \\ 0, & \text{otherwise} \end{cases}$$

The variables x_{it} will be referred as assignment variables (assigning nodes to time periods) and the variables z_{ijt} as transition variables (defining the path). With the use of these definitions, the following formulation for the TDTSP can be derived (Wiel and Sahinidis, 1996).

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{t=1}^{n} c_{ijt} z_{ijt} \qquad \text{(II.29)}$$

s.t.

$$\sum_{t=1}^{n} x_{it} = 1, \qquad i = 1,\dots,n \qquad \text{(II.30)}$$

$$\sum_{i=1}^{n} x_{it} = 1, \qquad t = 1,\dots,n \qquad \text{(II.31)}$$

$$\sum_{i=1}^{n} z_{ijt} = x_{jt}, \qquad j,t = 1,\dots,n \qquad \text{(II.32)}$$

$$\sum_{j=1}^{n} z_{ijt} = x_{i,t-1}, \qquad i,t = 1,\dots,n \qquad \text{(II.33)}$$

$$x_{it} \in \{0,1\}, \qquad i,t = 1,\dots,n \qquad \text{(II.34)}$$

$$z_{ijt} \in \{0,1\}, \qquad i,t = 1,\dots,n \qquad \text{(II.35)}$$

Constraints (II.30) and (II.31) give rise to a one-to-one mapping between cities and time periods. Thus, subtours elimination is also satisfied. Constraint (II.32) and (II.33) equate the number of transitions to and from each city, respectively, with its assignment in a feasible solution. (Wiel and Sahinidis, 1996) applied a Benders decomposition to this formulation in order to speed up lower bound calculation and then by embedding their bound in a branch and bound algorithm they solved pure time dependent traveling salesman instances having up to 18 nodes.

(Wiel and Sahinidis, 1995) presents an improvement heuristic for the TDTSP and a method for generating TDTSP instances with known optimal solutions.

Recently, (Bigras et al., 2006) consider single machine scheduling problems in sequence dependent setup environments and show how these problems can be seen as a special case of the TDTSP. They observed that the above formulation gave poor lower bounds and using such weak lower bounds in a branch and bound algorithm generally leads to large search trees. Stronger lower bounds for the TDTSP formulation are obtained by using cutting plane techniques of integer programming theory such as subtour cuts in the Dantzig-Fulkerson-Johnson formulation of TSP, 2-matching cuts and clique inequality in the node packing problem. Thus, they illustrate the benefits of using stronger formulations, even though the computation of their LP relaxation is more time consuming. Incorporating these improved LP relaxation bounds, an exact branch-and-bound algorithm capable of solving instances having 50 nodes is presented.

II.4.5. Exact algorithms

An algorithm is a set of fixed computational rules for solving a particular class of problems or models. Exact algorithms are guaranteed to find an optimal solution and to prove its optimality for every instance of a class of combinatorial optimization problems. The run-time, however, often increases dramatically with the instance size, and often only small or moderately sized instances can be practically solved to provable optimality. There does not exist an exact algorithm that solves all types of combinatorial optimization problems. However, several exact algorithms are developed for each model

II.4.5.1. Simplex algorithm

Simplex algorithm was developed by (Dantzig, 1951). In LP problems which consist of only two decision variables, the graphical method is used. Graphical method demonstrates that the optimal solution is always associated with a corner point of the solution space and development of simplex is based on this fact. Hence, the idea of the simplex algorithm is to move the solution to a new corner that has the potential to improve the value of the objective function in each iteration. The process terminates when the optimal solution is found (Taha, 2003). It is proven to be highly efficient in practice for solving LP problems, and therefore widely adopted in the commercial optimization packages for solving any LP model.

II.4.5.2. Interior Point methods

The simplex algorithm was for many years the main method used to solve LP problems. Over the last twenty years, however, the interior point algorithm has been developed. The simplex algorithm searches for the optimal solution along the corner points of the solution space, whereas the interior point algorithm looks for the optimum through the interior of the feasible region (Jensen and Bard, 2003). From the start it showed great theoretical promise, and considerable research in the area resulted in practical implementations that performed competitively with the simplex algorithm. More recently, interior point algorithms have evolved to become superior to the simplex algorithm, in general, especially when the problems are large. Among these interior point algorithms, Karmarkar's algorithm attracts attention because it provides a bound on the computational effort required to solve a problem that is a polynomial function of its size (Karmarkar, 1984).

II.4.5.3. Branch-and-Bound

Branch-and-bound is by far the most widely used tool for solving large scale IP and NLP problems. Branch-and-bound is however, an algorithm paradigm, which has to be filled out for each specific problem type and numerous choices for each of the components exist (Clausen, 1999). A Branch-and-bound algorithm searches the complete space of solutions for a given problem for the best solution. However, explicit enumeration is normally impossible due to exponentially increasing number of potential solutions. The use of bounds for the function to be optimized combined with the value of the current best solution enables the algorithm to search parts of the solution space only implicitly. It starts with solving an IP model as LP model by relaxing the integrality constraint. In case, the resultant LP solution or the continuous optimum is integer, this solution will also be the integer optimum. Otherwise, the branch-and-bound algorithm sets up lower and upper bounds for the optimal solution. The branching strategy repetitively decreases the upper bound and increases the lower bound for a minimization problem. The process terminates provided that the processing list is empty. Main principles of branch-and-bound algorithm, illustration of the method and different design issues through the Traveling Salesman problem, Graph Partitioning problem and Quadratic Assignment problem are analyzed in (Clausen, 1999).

II.4.5.4. Cutting plane methods

Cutting plane methods are exact algorithms for integer programming problems. They have proven to be very useful computationally in the last ten years, especially when combined with a branch and bound algorithm in a branch and cut framework. These methods work by solving a sequence of linear programming relaxations of the integer programming problem. The relaxations are gradually improved to give better approximations to the integer programming problem, at least in the neighborhood of the optimal solution. For hard instances that cannot be solved to optimality, cutting plane algorithms can produce approximations to the optimal solution in moderate computation times, with guarantees on the distance to optimality (Mitchell, 2002). A survey of applications of cutting plane methods, as well as a guide to the successful implementation of a cutting plane algorithm can be found in (Jünger et al., 1995). Cutting plane algorithms for general integer programming problems were first proposed by (Gomory, 1958, Gomory, 1963). The Gomory cuts were observed to give poor performance for large scale problems and were neglected for many years. Recently, (Balas et al., 1996a, Letchford and Lodi, 1992) showed that the Gomory cuts can actually be useful. There has been interest recently in other families of cutting planes for IP problems. Two such families are lift-and-project cuts and Fenchel cuts (Balas et al., 1996b, Boyd, 1994). A recent study by (Marchand et al., 2002) is a good survey that presents cutting planes that are useful or potentially useful in solving mixed integer programs.

II.4.5.5. Branch-and-cut algorithms

Branch-and-cut algorithms are also exact algorithms consisting of a combination of a cutting plane method with a branch-and-bound algorithm. The method solves the LP relaxation of IP problem using simplex algorithm. When an optimal solution is obtained, and this solution has a non-integer value for a variable that is supposed to be integer, a cutting plane algorithm is used to find further linear constraints which are satisfied by all feasible integer points but violated by the current fractional solution. If such an inequality is found, it is added to the linear program, such that resolving it will yield a different solution which is hopefully "less fractional". This process is repeated until either an integer solution is found (which is then known to be optimal) or until no more cutting planes are found.

At this point, the branch and bound part of the algorithm is started. The problem is split into two versions, one with the additional constraint that the variable is greater than or equal to the next integer greater than the intermediate result, and one where this variable is less than or equal to the next lesser integer. In this way new variables are introduced in the basis according to the number of basic variables that are non-integers in the intermediate solution but which are integers according to the original constraints. The new linear programs are then solved using the simplex method and the process repeats until a solution satisfying all the integer constraints is found.

One aspect of a branch-and-cut approach that should not be overlooked is that it can be used to provide bounds. In particular, if we are minimizing but we are unable to prove optimality, a lower bound on the optimal value can be deduced from the algorithm, which can be used to provide a guarantee on the distance from optimality. Therefore, for large and/or hard problems, branch-and-cut can be used in conjunction with heuristics or metaheuristics to obtain a good (possibly optimal) solution and also to indicate how far from optimality this solution may be (Mitchell, 2002).

Padberg and Rinaldi introduce a branch-and-cut algorithm for solving large scale instance of TSP to optimality (Padberg and Rinaldi, 1991). Reference (Jünger et al, 1995) is good for cutting plane algorithms and branch-and-cut algorithms from an implementer’s point of view. Recently, Mitchell describes how a branch-and-cut algorithm can be tailored to a specific IP problem, and how families of general cutting planes can be used to solve a wide variety of problems (Mitchell, 2002).

II.4.6. Heuristics and Meta-Heuristics

As stated previously, exact algorithms are guaranteed to find an optimal solution and to prove its optimality for every instance of a class of combinatorial optimization problems. The run-time, however, often increases dramatically with the instance size, and often only small or moderately sized instances can be practically solved to provable optimality. In this case, the only possibility for larger instances is to trade optimality for run-time, yielding heuristic algorithms. In other words, the guarantee of finding optimal solutions is sacrificed for the sake of getting good solutions in a limited time. The term ‘heuristic’ is referred to the methods for the problem under study, based on rule of thumb, common sense, or adaptations of exact methods for simpler models. They are used to find reasonable solutions when the problems are complex and difficult to solve. In optimization, a heuristic method refers to a practical and quick method based on strategies that are likely to (but not guaranteed to) lead a solution that is approximately optimal or near-optimal.

Heuristics can be classified as either the constructive heuristics or as the local search heuristics. First, the constructive heuristics (such as the nearest neighbor heuristic for TSP) simply applies a rule of thumb at each step and builds the solution. Constructive heuristics are designed according to the special status of the problem to be solved. For example, the simplest constructive heuristic for TSP obtains the tour by building the Minimum Spanning Tree whereas the best-known approach to the VRP problem is the "savings" algorithm of Clarke and Wright. These constructive heuristics can never guarantee to find optimality in every instance of the problem but for most of them an upper bound analysis can be made indicating how much the result is far from optimum.

Local search heuristics are based on the concept of exploring the vicinity of the current solution. Neighboring solutions are generated by a move generation mechanism. If the generated neighbor has a better objective value, it becomes a new current solution, or otherwise the current solution is retained. The process is iterated until there is no possibility of improvement in the neighboring solution. Hence, the method may stop at a local optimum which is far away from the global optimum. In order to avoid getting trapped at a local optimum, a number of conceptual strategies have been developed for the local search heuristics. These heuristics are referred to as the meta-heuristics (Osman and Kelly, 1996). A meta-heuristic is an iterative generation process which guides a subordinate or simple heuristic by combining intelligence, biological evolution, neural systems, and statistical mechanics, for exploring and exploiting the search spaces using learning strategies to structure information in order to find near-optimal solutions effectively.

The meta-heuristic notion comprises –but not restricted to – Ant Colony Optimization (ACO), Evolutionary Computing including Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS).

II.4.6.1. Simulated Annealing

Simulated Annealing is commonly said to be the oldest among the meta-heuristics and surely one of the first algorithms that had an explicit strategy to escape from local minima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (uphill moves) in order to escape from local minima. The probability of doing such a move is decreased during the search. The algorithm starts by generating an initial solution (either randomly or heuristically constructed) and by initializing the so-called temperature parameter *T*. Then, at each iteration a solution *s'* in *N(s)* is randomly sampled and it is accepted as new current solution depending on *f(s)*, *f(s')* and *T*. *s'* replaces *s* if $f(s') < f(s)$ or, in case $f(s') \geq f(s)$, with a probability which is a function of *T* and $f(s')-f(s)$. The probability is generally computed following the Boltzmann distribution $e^{-\frac{f(s')-f(s)}{T}}$ (Blum and Roli, 2003).

II.4.6.2. Tabu Search

The Tabu Search algorithm applies a best improvement local search as basic ingredient and uses a short term memory to escape from local minima and to avoid cycles. The short term memory is implemented as a tabu list that keeps track of the most recently visited solutions and forbids moves toward them. The neighborhood of the current solution is thus restricted to the solutions that do not belong to the tabu list, which we call as allowed set in the following. At each iteration the best solution from the allowed set is chosen as the new current solution. Additionally, this solution is added to the tabu list and one of the solutions that were already in the tabu list is removed and usually first in first out rule is applied. The algorithm stops when a termination condition is met. It might also terminate if the allowed set is empty, that is if all the solutions in *N(s)* are forbidden by the tabu list (Blum and Roli, 2003).

II.4.6.3. Genetic Algorithms

Genetic algorithms are population based methods inspired by the principles of natural evolution (Holland, 1990). The algorithm starts the search with a population of individual chromosomes (solutions) generated randomly or heuristically. In each iteration, the population is evolved using genetic operators such as mutation and crossover to produce offspring (new individuals of the next generation). Mutation is unary operator that introduces random modifications of the chromosome in order to add diversity to the population. The crossover operator combines two parents (individuals from the current generation) to generate new offspring. The crossover operation aims to propagate good solution components from parents to offspring. The selection mechanism chooses the parents based on survival of the fittest. That is, the better fitness values are more likely to be chosen to undergo reproduction in order to produce offspring (Beasley et al., 1993).

CHAPTER III

OPTIMIZING THE ASSEMBLY OPERATIONS OF CHIP MOUNTER PLACEMENT MACHINES AND THE SEQUENCE DEPENDENT TSP

This chapter gives the formulation of a new generalization of the TSP, formulation of problems emerging from the operations of a chip mounter machine and the details of the work done for optimizing the operations of this machine. First section gives the formulation of a new generalization of the TSP. This new generalization arises when the operations of chip mounter and chip shooter machines are analyzed. Section two gives the formulation of the placement sequencing problem which make use of new the TSP generalization introduced in section one. Section two also includes the formulation of feeder configuration and integrated problems. Heuristics for improving the efficiency of a chip mounter machine are given in the third section.

III.1. SEQUENCE DEPENDENT TSP: A NEW GENERALIZATION OF THE TSP

In this section, a new generalization of the TSP will be introduced to the literature. This new generalization is observed in analyzed machines in this thesis.

In the classical TSP, cost of travel between two cities depends only on the distance between them. If Chebyshev measure is used in Cartesian coordinate system, then distance between two points *i* and *j*, *d(i, j)*, is defined as max$\{|i_x-j_x|, |i_y-j_y|\}$, where i_x and i_y represent the x and y-coordinates of point *i*, respectively. In this study, we calculate cost in time units, hence we define the cost function as follows.

$$CF1(i,j) = \frac{d(i,j)}{v} \tag{III.1}$$

where *v* is a predefined speed value.

If we define a new cost function *CF2*, then the problem gets a bit more complicated.

$$CF2(i, j, ft_j)=max(CF1(i, j), ft_j) \qquad \text{(III.2)}$$

where ft_j is a predefined free time value for each point j. It is called as free time since during this time interval the salesman is free to move to any vertex j without increasing the cost. So, the cost function depends not only on the distance between them but also the free time value of the destination point. Actually, this problem turns out to be the Asymmetric TSP (ATSP) because $CF2(j, i, ft_i) \neq CF2(i, j, ft_j)$, unless $ft_j= ft_i$. Furthermore, if free time values for all points are constant for all points, then the problem turns out to be a classical TSP with Chebyshev distance measure.

In the generalization of the problem that we introduce, cost of travel between two cities depends on the distance between these cities, on the free time value of the destination city and also free time value of a number of following cities (say *d*). So, the position of each city becomes important because it can affect the cost calculation of travel between former cities. Hence, the problem turns out to be a more complex then both ATSP and TDTSP. We will call this problem as Sequence Dependent TSP (SDTSP), because the cost calculations are dependent on the travel sequence.

So, for a given travel sequence, cost of travel between points i and j when point j is visited in p^{th} position is calculated by

$$CF3(i, j, p)=max(CF1(i, j), max(ft_p, ft_{p+1}, ft_{p+2}, \ldots, ft_{p+d}))= \max\left(CF1(i,j), \max_{m=p}^{p+d}\left(ft_m\right)\right) \quad \text{(III.3)}$$

where ft_{p+1} is the free time of the point visited in $(p+1)^{th}$ position, ft_{p+2} is the free time of the point visited in $(p+2)^{th}$ position, and so on. d represents the number of points in the travel sequence whose free time characteristic will be taken into account in the cost calculation. If d is zero, then *CF3* turns out to be $max(CF1(i, j), ft_j)$ which is equal to *CF2* and thus the problem becomes an ATSP as shown above. Besides the former condition, if cost of travel from point i to j does not depend on free time but on its position in the travel sequence, then problem turns out to be a TDTSP.

Below, the mathematical representation of the SDTSP will be given. Firstly, the notation used for the problem should be given.

N: total number of points to be visited,

p: is the visiting order or visiting position.

d: number of cities whose free time must be taken into account beforehand while travelling to a city

We also need to define decision variable x_{ip} which denote the assignment of node *i* to position *p*.

$$x_{ip} = \begin{cases} 1, & \text{if node i is visited in p}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

It is important to note a detail about indices used in formulations below. We assume that travelling along the path restarts after a tour ends. Hence, in the formulations if an index exceeds its upper limit, it should be considered as if the counting of indices restart from one. For example if we are talking about $(p+d)^{th}$ visit when *p* is 98, *d* is five and *N* is 100, then $(p+d)^{th}$ visit means the third point of the next tour.

Then, cost of travel from point *i* to point *j* when point *j* is visited in p^{th} position, C_{ijp} is

$$C_{ijp} = x_{i,p-1} x_{jp} \max\left(CF1(i,j), \max_{m=p}^{p+d}\left(\sum_{l=1}^{N} ft_l x_{lm} \right)\right) \qquad \text{(III.4)}$$

Since determining the travel sequence is the objective of the problem, we defined C_{ijp} by incorporating the decision variables x_{ip} into *CF3*. In this formulation, first term in outer max function is the time for traveling from *i* to *j*. The second term finds the maximum of free time values of the following *d +1* points to be visited, including point *j*. The multiplication term calculates the free time for the point visited in m^{th} position where *m* states the positions from *p* to *p+d*. Hence, cost of travel from point *i* to point *j* is nonzero if point *i* is visited in *(p-1)*th position and point *j* is visited in p^{th} position.

Then the mathematical model for the problem can be stated as follows.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{p=1}^{N} C_{ijp} \qquad \text{(III.5)}$$

s.t.

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\dots,N \tag{III.6}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\dots,N \tag{III.7}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\dots,N \tag{III.8}$$

Constraint (III.6) guarantees that each node is assigned to a position and constraint (III.7) guarantees that each position is occupied by only one node. There is no need for the constraints of the classical TSP that are used to prevent subtour elimination, because placing each node in exactly one position is achieved by these constraints.

The problem may seem easy with this formulation but it is nonlinear because of maximum functions used in C_{ijp} definition and quadratic term in the objective function. In order to remove the nonlinearity in the objective function, we firstly introduce D_{ijp} as the dominating factor in C_{ijp} definition.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{III.9}$$

s.t.

$$D_{ijp} - x_{i,p-1} x_{jp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.10}$$

$$D_{ijp} - x_{i,p-1} x_{jp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.11}$$

$$D_{ijp} - x_{i,p-1} x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.12}$$

$$\vdots$$

$$D_{ijp} - x_{i,p-1} x_{jp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.13}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\dots,N \tag{III.14}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\dots,N \qquad \text{(III.15)}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\dots,N \qquad \text{(III.16)}$$

$D_{ijp} \geq 0$ and integer (III.17)

This formulation has now a linear objective function, but it still has nonlinear terms in constraints.

The quadratic term $x_{i,p-1}x_{jp}$ in (III.10-III.13) can be eliminated by applying the following rules (Williams, 1999):

- Replace the product term xy by a binary variable w.
- Impose the logical condition w=1 if and only if x=1 and y=1 by means of the extra constraints $w \leq x$, $w \leq y$ and $w \geq x+y-1$.

Hence, we introduce the variable w_{ijp} with the following interpretation.

$$w_{ijp} = \begin{cases} 1, & \text{if node j is visited in p}^{\text{th}} \text{ position after node i was visited in (p - 1)}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

Then the problem can be stated as

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \qquad \text{(III.18)}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\dots,N \qquad \text{(III.19)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0 \qquad i,j,p = 1,\dots,N \qquad \text{(III.20)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0 \qquad i,j,p = 1,\dots,N \qquad \text{(III.21)}$$

$\vdots$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0 \qquad i,j,p = 1,\dots,N \qquad \text{(III.22)}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\dots,N \qquad \text{(III.23)}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\dots,N \qquad \text{(III.24)}$$

$$w_{ijp} \le x_{i,p-1}, \qquad i,j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.25)}$$

$$w_{ijp} \le x_{jp}, \qquad i,j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.26)}$$

$$w_{ijp} \ge x_{i,p-1} + x_{jp} - 1, \qquad i,j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.27)}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\dots,N \qquad \text{(III.28)}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\dots,N \qquad \text{(III.29)}$$

D_{ijp} ≥0 and integer (III.30)

Constraints (III.25-III.27) can be stated more compactly and interpretably after the following mathematical discussion. If we sum up the terms in (III.25-III.27) across the *i* index, we obtain

$$\sum_{i=1}^{N} w_{ijp} \le \sum_{i=1}^{N} x_{i,p-1}, \qquad j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.31)}$$

$$\sum_{i=1}^{N} w_{ijp} \le \sum_{i=1}^{N} x_{jp}, \qquad j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.32)}$$

$$\sum_{i=1}^{N} w_{ijp} \ge \sum_{i=1}^{N} x_{i,p-1} + \sum_{i=1}^{N} x_{jp} - N, \qquad j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.33)}$$

Equation (III.31) can be rewritten by using (III.24) as follows

$$\sum_{i=1}^{N} w_{ijp} \le 1, \qquad j,p = 1,\dots,N,\ \ i \ne j \qquad \text{(III.34)}$$

If x_{jp}=1, then by using Equations (III.33) and (III.34) we can say that

$$1 \le \sum_{i=1}^{N} w_{ijp} \le 1, \quad j,p = 1,\dots,N,\ \ i \ne j \Leftrightarrow \sum_{i=1}^{N} w_{ijp} = 1, \quad j,p = 1,\dots,N,\ \ i \ne j \quad \text{(III.35)}$$

If x_{jp}=0 then, by using Equations (III.29) and (III.32) we can conclude that

$$\sum_{i=1}^{N} w_{ijp} \le 0, \quad j,p = 1,\dots,N,\ \ i \ne j \Leftrightarrow \sum_{i=1}^{N} w_{ijp} = 0, \quad j,p = 1,\dots,N,\ \ i \ne j \quad \text{(III.36)}$$

Hence, we can conclude that

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j, p = 1, \ldots, N \qquad \text{(III.37)}$$

After summing up the terms in (III.25)-(III.27) across the *j* index and carrying out a similar discussion as above, we can easily conclude that

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i, p = 1, \ldots, N \qquad \text{(III.38)}$$

So, constraints (III.25)-(III.27) can be stated by the equations (III.37) and (III.38).

Why we transformed the equations into a different formulation is because we want to interpret these constraints more explicitly according to our context.

Constraint (III.37) states that if point *j* is assigned to *p*th position, then exactly one point precedes *j* in the travel sequence. Similarly (III.38) states that if point *i* is assigned to (*p-1*)$^{\text{th}}$ position then exactly one point succeeds it in the travel sequence.

The complete nonlinear integer programming formulation of SDTSP is as follows.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \qquad \text{(III.39)}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i, j) \geq 0 \qquad i, j, p = 1, \ldots, N \qquad \text{(III.40)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{lp} \geq 0 \qquad i, j, p = 1, \ldots, N \qquad \text{(III.41)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{l,p+1} \geq 0 \qquad i, j, p = 1, \ldots, N \qquad \text{(III.42)}$$

$$\vdots$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} ft_l x_{l,p+d} \geq 0 \qquad i, j, p = 1, \ldots, N \qquad \text{(III.43)}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N \qquad \text{(III.44)}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N \qquad \text{(III.45)}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\ldots,N \qquad \text{(III.46)}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\ldots,N \qquad \text{(III.47)}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\ldots,N \qquad \text{(III.48)}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\ldots,N \qquad \text{(III.49)}$$

$$D_{ijp} \geq 0 \text{ and integer} \qquad \text{(III.50)}$$

In the next section, formulation of problems of chip mounter machine will be given.

III.2. PROBLEM FORMULATION OF CHIP MOUNTER PLACEMENT MACHINE

This section initially gives definitions and notation to be used in the following formulations. Afterwards, the nonlinear integer programming formulation of the placement sequencing problem and linear integer programming formulation of feeder configuration problem will be given. Then, the assembly time minimization problem for the chip mounter machine which is comprised of these two problems and its nonlinear integer programming formulation will also be given.

III.2.1. Definitions and Notation

In previous chapters the chip mounter placement machine is described in detail along with its operation principles. It is also mentioned that there are two problems that must be defined, formulated and solved for this machine. First one is the placement sequencing problem and the other one is the feeder configuration problem. Actually, the assembly time minimization problem for the chip mounter machine is comprised of these two problems and its formulation should also be given.

Before defining the problems, we should give the notation used for them. In this notation, machine specific values are also stated by generic notation in order to express a generic formulation. For example number of weight categories for component types is four in our case but is expressed by *K* below.

N: total number of components to be placed,

n: number of component types

K: number of weight categories (*K*=4 in our case)

N_k: number of components in each weight category *k*, *k=1,2,…,K*.

n_k: number of component types in each weight category *k*, *k=1,2,…,K*.

$$\sum_{k=1}^{K} n_k = n$$

R: number of feeder slots (*R*=60, but no components are assigned to first 20 slots because they correspond to no pickup zone, *npz*)

H: number of heads (*H*=72 in our case)

npz: number of heads in no pickup zone (*npz*=20 in our case)

ct_{it}: component type matrix (*N x n*) indicating the component type for each component

$$ct_{it} = \begin{cases} 1 & \text{if component } i \text{ is of component type } t \\ 0 & \text{otherwise} \end{cases}$$

g_{tk}: group (weight category) matrix (*n x K*) indicating the group for each component type

$$g_{tk} = \begin{cases} 1 & \text{if component type } t \text{ is in group } k \\ 0 & \text{otherwise} \end{cases}$$

tt_k : turret rotation time for each weight category *k*, (When *k=1*, the turret rotation speed is the maximum, that is, turret rotation time is minimum)

p: is the placement order or placement position.

In order to express the placement sequence we need to define decision variable x_{ip} which denote the assignment of node *i* to position *p*.

$$x_{ip} = \begin{cases} 1, & \text{if node i is visited in p}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

Decision variable w_{ijp} is used for expressing the travel from node *i* visited in (*p-1*)*th* position to node *j* visited in *p*th position.

$$w_{ijp} = \begin{cases} 1, & \text{if node j is visited in p}^{\text{th}} \text{ position after node i was visited in (p - 1)}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

On the other hand, in order to express the feeder configuration, we need to define decision variable y_{tr} to state the status of assigning component type t to feeder r.

$$y_{tr} = \begin{cases} 1, & \text{if component type } t \text{ is stored in feeder } r \\ 0, & \text{otherwise} \end{cases}$$

r is the feeder number where component magazine is placed behind heads 21-60 and a feeder slot corresponds to each head. For simplifying the formulations below, we assume that component magazine stand along heads 1-60 but no component types are assigned to feeder slots between one and 20 because they correspond to no pickup zone. So, $r \in \{1,\ldots,60\}$, however first 20 columns in the y_{tr} array are always set to zero.

Then, let us focus on placement sequencing problem by assuming that the feeder configuration is given.

III.2.2. Placement Sequencing Problem

Given a feeder configuration, the objective of the placement sequencing problem is to find a placement sequence of the components so that the assembly process is completed in the shortest time possible. In order to express placement sequencing problem, we need to define time between the completion of consecutive placements at points x and y. The time between consecutive placements at points x and y is defined by *CF2*.

Recall that *CF2* takes the maximum of two values and the analyzed machine has two simultaneously operating parts and in order to complete a placement operation the two parts must finish their operations. The first part is the rotational turret and tt_k denotes the time it needs to complete its operation. Formally, we define turret time, tt_k, as the turret rotation time required for the next placement head to arrive over the PCB. Turret time is also known as the free time since during this time interval the board carrier is free to move to any placement point without increasing the cost. In other words, the points at a Chebyshev distance of free time or less can be taken as equidistant points. When the component types to be placed are categorized according to their weight, we observe that there are four types of component types to be placed on the PCB. The speed of the turret is determined according to the heaviest component type carried. Hence, tt_k takes four values such that tt_1=0.20 s, tt_2=0.23 s, tt_3=0.33 s and tt_4=0.40 s. The second part of the machine is board carrier and the time for board carrier to align the PCB under the placement head of the turret is calculated by *CF1* as explained in (III.1).

If the rotational turret rotates in a unique speed value, then the problem turns out to be a classical TSP with Chebyshev measure. Thus, what makes the problem complicated is the varying rotational speed of the turret and this property yields the SDTSP.

In order to further clarify how a complex problem arises, we need to make the following discussion. In our case, turret time can take four possible values where its value is determined by the heaviest component type being carried by one of the heads of the turret. In other words, for the placement of component *j* on head number 1, the rotational speed value corresponding to the heaviest component being carried by the heads numbered 1–60 will be the current turret time value for this rotational step of the turret. Cost of travel between two vertices *i* and *j* depends on vertice *j*'s and following up to 59 components' free time value in the travel sequence. In such a case, cost calculation can be stated by *CF3* (Equation III.3) and thus, the problem turns out to be an SDTSP as we explain below.

We should remind that in the formulations below, if an index exceeds its upper limit, then it should be considered as if the counting of indices restart from one, due to continuous production.

Then, cost of travel from component *i* to component *j* when component *j* is assembled in p^{th} position, C_{ijp} is

$$C_{ijp} = w_{ijp} \max\left(CF1(i,j), \max_{m=0}^{R-1}\left(\sum_{l=1}^{N}\sum_{t=1}^{n}\sum_{k=1}^{K}\sum_{u=m+1}^{R} tt_k g_{tk} ct_{lt} x_{l,p+m} y_{tu} \right)\right) \qquad \text{(III.51)}$$

In this formulation, first term in outer max function is the time for board carrier to align the PCB under the placement head of the turret. The second term finds the maximum of turret time values of the following components that are carried by any of the heads, including component *j*. More explicitly, for calculating the placement time for component *j* (placed in p^{th} position), we need to answer the following question; Is the component to be placed *v* steps later, (call it component *l*), currently being carried by one of the heads or not. To answer this question, all feeder slots within indices from v to R are scanned. If the component type of component l is stored in one of these scanned slots, this means that component l is being carried by one of the heads and its turret time should be taken into consideration. Otherwise, component l is not picked up yet, hence it should be neglected in the placement time calculation. This question should be answered for R components to be placed after component j and maximum turret time of those carried components effects the placement time. Shortly, the multiplication term calculates the turret time for the component to be placed in the following positions, if they are being carried by the heads.

Observe that (III.51) is actually the cost function given by (III.3) that constitutes the basis of SDTSP formulation. If we define the free time value for component l as follows, then the problem turns out to be the SDTSP.

$$ft_l = \sum_{t=1}^{n}\sum_{k=1}^{K} tt_k g_{tk} ct_{lt} \qquad \text{(III.52)}$$

After removing the nonlinearity in C_{ijp} by firstly introducing D_{ijp} as the dominating factor in C_{ijp} definition, we obtain the following nonlinear integer programming formulation of the problem.

$$\min \sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{p=1}^{N} D_{ijp} \qquad \text{(III.53)}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.54)}$$

$$D_{ijp} - w_{ijp}\sum_{l=1}^{N}\sum_{t=1}^{n}\sum_{k=1}^{K}\sum_{u=1}^{R} tt_k g_{tk} ct_{lt} x_{lp} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.55)}$$

$$D_{ijp} - w_{ijp}\sum_{l=1}^{N}\sum_{t=1}^{n}\sum_{k=1}^{K}\sum_{u=2}^{R} tt_k g_{tk} ct_{lt} x_{l,p+1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.56)}$$

$$\vdots$$

$$D_{ijp} - w_{ijp}\sum_{l=1}^{N}\sum_{t=1}^{n}\sum_{k=1}^{K}\sum_{u=R}^{R} tt_k g_{tk} ct_{lt} x_{l,p+R-1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.57)}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N, \qquad \text{(III.58)}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N, \qquad \text{(III.59)}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j, p = 1,\ldots,N, \qquad \text{(III.60)}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i, p = 1,\ldots,N, \qquad \text{(III.61)}$$

$$x_{ip} \in \{0,1\}, \qquad i, p = 1,\ldots,N, \qquad \text{(III.62)}$$

$$w_{ijp} \in \{0,1\}, \qquad i, j, p = 1,\ldots,N, \qquad \text{(III.63)}$$

We introduce several constraint sets to the problem. Constraint set (III.54) imposes the requirement that placement time for component *j* (component *j* being placed after component *i* in *p*th position) cannot be smaller than the travel time from *i* to *j*. Constraint sets (III.55-III.57) (representing *R* different constraint sets) impose that placement time for component *j* cannot be smaller than the turret time values associated with the incoming components for placement, if they are already picked up by the heads. Constraint (III.58) guarantees that each component is assigned to a position and constraint (III.59) guarantees that each position is occupied by only one node. There is no need for constraints that are used to guarantee subtour elimination, because placing each node in exactly one position is achieved by these constraints. Constraint (III.60) states that if component *j* is assigned to *p*th position, then exactly one component precedes *j* in the travel sequence. Similarly (III.61) states that if component *i* is assigned to (*p-1*)th position then exactly one component precedes it in the travel sequence. Constraints (III.62) and (III.63) are the binary constraints for decision variables x_{ip} and w_{ijp}.

Thus our formulation for the placement sequencing problem is complete. Feeder configuration problem will be formulated in the next subsection.

III.2.3. Feeder Configuration Problem

Given a placement sequence, the ultimate objective of feeder configuration problem is to minimize total assembly time. Hence, the objective function and its constraints defined for achieving its linearity while formulating the placement sequencing problem can be exactly used in the formulation of feeder configuration problem.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{III.64}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \tag{III.65}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=1}^{R} tt_k g_{tk} ct_{lt} x_{lp} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{III.66}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=2}^{R} tt_k g_{tk} ct_{lt} x_{l,p+1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{III.67}$$

$$\vdots$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=R}^{R} tt_k g_{tk} ct_{lt} x_{l,p+R-1} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{III.68}$$

$$\sum_{r=npz+1}^{R} y_{tr} = 1, \qquad t = 1,\ldots,n, \tag{III.69}$$

$$\sum_{t=1}^{n} y_{tr} = 1, \qquad r = npz+1,\ldots,R, \tag{III.70}$$

$$y_{tr} = 0, \qquad t = 1,\ldots,n, \qquad r = 1,\ldots,npz \tag{III.71}$$

$$y_{tr} \in \{0,1\}, \qquad t = 1,\ldots,n, \qquad r = npz+1,\ldots,R, \tag{III.72}$$

Constraint (III.69) guarantees that each component type is assigned to a feeder slot and constraint (III.70) guarantees that each slot is occupied by only one component type. We assign no component types to the slots corresponding to no pickup zone by constraint (III.71).

Actually, feeder configuration problem is a linear assignment problem with additional linear constraints. These constraints are added for the sake of obtaining a linear objective function.

Now, the formulations of both problems are complete. However, placement sequencing and feeder configuration problems are interdependent. This dependency can be depicted as follows. We use x_{ip} variables in feeder configuration problem after they are solved in placement sequencing problem. On the other hand, y_{tr} values are used in placement sequencing problem whereas they are calculated in feeder configuration problem.

Next, we will give the formulation of assembly time minimization problem which is the combination of placement sequencing and feeder configuration problems.

III.2.4. Assembly time minimization problem

Recall that, the ultimate objective of the optimization problem is to minimize the assembly time. That is, the objective is to find a placement sequence of the components and a feeder configuration so that the assembly process is completed in the shortest time possible. Hence, we build the following nonlinear integer programming that searches for optimum values of x_{ip} and y_{tr} variables simultaneously.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \qquad \text{(III.73)}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.74)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=1}^{R} tt_k g_{tk} ct_{lt} x_{lp} y_{tu} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(III.75)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=2}^{R} tt_k g_{tk} ct_{lt} x_{l,p+1} y_{tu} \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.76}$$

$$\vdots$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=R}^{R} tt_k g_{tk} ct_{lt} x_{l,p+R-1} y_{tu} \geq 0 \qquad i,j,p = 1,\dots,N \tag{III.77}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\dots,N, \tag{III.78}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\dots,N, \tag{III.79}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\dots,N, \tag{III.80}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\dots,N, \tag{III.81}$$

$$\sum_{r=npz+1}^{R} y_{tr} = 1, \qquad t = 1,\dots,n, \tag{III.82}$$

$$\sum_{t=1}^{n} y_{tr} = 1, \qquad r = npz+1,\dots,npz+R, \tag{III.83}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\dots,N, \tag{III.84}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\dots,N, \tag{III.85}$$

$$y_{tr} = 0, \qquad t = 1,\dots,n, \qquad r = 1,\dots,npz \tag{III.86}$$

$$y_{tr} \in \{0,1\}, \qquad t = 1,\dots,n, \qquad r = npz+1,\dots,npz+R, \tag{III.87}$$

There is no need to explain the constraints written above because they are all explained in previous subsections.

III.2.5. Secondary Problem

Besides, as the future work of his study, Duman directs us for improving the assembly time by altering the placement sequence of components within lightest weight category route (group-1 components) (Duman, 2007a). The following example clarifies the situation. Consider a PCB where 100 components must be placed and let 80, 10, 5 and 5 components are in groups 1, 2, 3 and 4, respectively. Applying the formula of SC_i calculation for ATMA given in the referred study, turret time value is 0.20 s for the first 50 placements, 0.23 s for the next 15 placements, 0.33 s for the next 10 placements and 0.40 s for the next 25 placements. Note that, even for the placement of 80 components in group-1, there are four different values of tt_k. This variability in the tt_k values raises the question: Of the 80 components in group-1 which 50, 15, 10 and 5 should be selected to be placed when turret time is 0.20, 0.23, 0.33 and 0.40 s, respectively.

This problem will be called secondary problem because, it comes out when ATMA is applied to the problem and its optimization affects the total assembly time at a secondary level when compared with the above formulated problems.

The formulation of secondary problem can be done as follows. The secondary problem is focused on determining the placement sequence of group-1 components. For a generic formulation we introduce the notation M_k to denote the number of points in group-1 to be visited when cost function is *CF2(x, y, tt_k)*. The values of M_k for each k can be easily calculated according to the N_k and SC_k values as done in the above example. Observe that $\sum_{k=1}^{K} M_k = N_1$. A calculation scheme for M_k values will be given in the next subsection.

$$\min \sum_i \sum_j \sum_p C_{ijp} w_{ijp} \tag{III.88}$$

s.t.

$$\sum_{p=1}^{N_1} x_{ip} = 1, \qquad i = 1,\ldots,N_1, \tag{III.89}$$

$$\sum_{i=1}^{N_1} x_{ip} = 1, \qquad p = 1,\ldots,N_1, \qquad \text{(III.90)}$$

$$\sum_{i=1}^{N_1} w_{ijp} = x_{jp}, \qquad j = 1,\ldots,N_1, \qquad p = 1,\ldots,N_1, \qquad \text{(III.91)}$$

$$\sum_{j=1}^{N_1} w_{ijp} = x_{jp}, \qquad i = 1,\ldots,N_1, \qquad p = 1,\ldots,N_1, \qquad \text{(III.92)}$$

C_{ijp}=$CF2(i, j, tt_1)$ for p=1,…, M_1, (III.93)

C_{ijp}=$CF2(i, j, tt_2)$ for p=M_1+1…, $M_1 + M_2$, (III.94)

C_{ijp}=$CF2(i, j, tt_3)$ for p=$M_1 + M_2$ +1,…, $M_1 + M_2 + M_3$, (III.95)

…

C_{ijp}=$CF2(i, j, tt_K)$ for p= $\sum_{k=1}^{K-1} M_k$ +1,…,N_1 (III.96)

$$x_{ip} \in \{0,1\}, \qquad i = 1,\ldots,N_1, \qquad p = 1,\ldots,N_1, \qquad \text{(III.97)}$$

$$w_{ijp} \in \{0,1\}, \qquad i = 1,\ldots,N_1, \qquad j = 1,\ldots,N_1, \qquad p = 1,\ldots,N_1, \qquad \text{(III.98)}$$

In this formulation, x_{ip} decision variables and w_{ijp} are used in the same way they are defined previously. This problem is actually a case of TDTSP with special cost functions. Constraints (III.89) and (III.90) guarantee that each node is assigned to only one position. Hence, subtours are not a problem in the formulation. Constraint (III.91) states that if point j is assigned to p^{th} position, then exactly one point precedes j in the travel sequence. Similarly (III.92) states that if point i is assigned to $(p\text{-}1)^{\text{th}}$ position then exactly one point proceeds it in the travel sequence. Cost of travel between two points i and j is calculated by considering its position in the travel sequence and it is predetermined by the constraints (III.93)-(III.96). Thus, the above formulation is the same as the TDTSP formulation given by (Wiel and Sahinidis, 1996).

III.3. OPTIMIZING THE ASSEMBLY OPERATIONS OF CHIP MOUNTER PLACEMENT MACHINES

The algorithm ATMA proposed for chip mounter machines is a very effective algorithm. This is because it takes care of placing components with different weights without reducing the turret rotation speed earlier than necessary. However, when its performance is compared with the lower bound value (assembly time value obtained when all of the feeder carriage and board carrier movements are completed within turret time), it is observed that much better improvements can be obtained by more effective heuristics (Duman, 2007a). Besides, the optimization problem defined as secondary problem looks promising for arising new improvement opportunities.

A common approach for solving combinatorial optimization problems is developing heuristic methods. In heuristic methods we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time. Thus, the use of heuristic methods to solve combinatorial optimization problems has received more and more attention in the last 30 years.

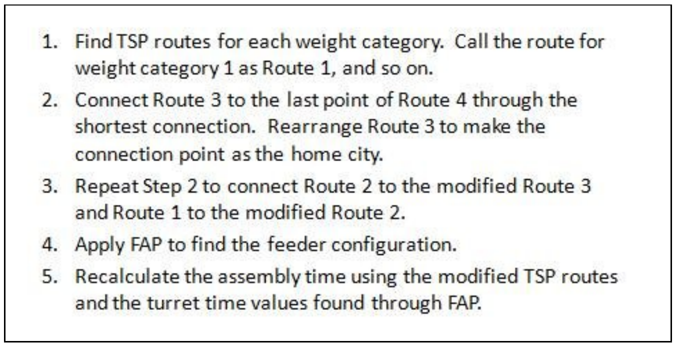
In the following subsections, the proposed heuristics will be explained in detail.

III.3.1. iATMA

In this thesis, we proposed an improved version of the ATMA algorithm called inverse ATMA (iATMA) and obtain further improvements by several additional heuristics. It is important to note that in the rest of the study the terms “point”, “vertex” and “component” are used interchangeably.

The idea behind iATMA is similar to ATMA in terms of the algorithmic steps. The basic difference that it proposes is in the order of placement of component groups. Both algorithms mount the components in groups of their weight categories. ATMA places the component groups starting from group-1 to group-4 (i.e. lightest to heaviest), but iATMA inverses this process such that it starts mounting the components from group-4 to group-1 (i.e. heaviest to lightest).

Algorithm iATMA is given in Figure III.1.



1. Find TSP routes for each weight category. Call the route for weight category 1 as Route 1, and so on.
2. Connect Route 3 to the last point of Route 4 through the shortest connection. Rearrange Route 3 to make the connection point as the home city.
3. Repeat Step 2 to connect Route 2 to the modified Route 3 and Route 1 to the modified Route 2.
4. Apply FAP to find the feeder configuration.
5. Recalculate the assembly time using the modified TSP routes and the turret time values found through FAP.

Figure III.1—Algorithm iATMA

We propose no new ideas for FAP because it already gives ideal solutions.

In order to see the improvement that iATMA provides, consider the following example. Let *N=100*, i.e. 100 components will be assembled with the following weight categories: $N_1=80$, $N_2=10$, $N_3=5$ and $N_4=5$ where increasing index value identifies heavier component categories. After configuring the feeder slots as stated in the FAP, using ATMA, it can be found that the number of rotational steps of the turret in each speed category (SC_i) as 50, 15, 10 and 25, respectively, if continuous assembly process of the PCB assembly is considered.

If the iATMA is to be used in this example, the number of rotational steps of the turret in each speed category would be 60, 10, 5 and 25.

Calculation scheme for the number of rotational steps of the turret in each speed category *i*, (SC_i) for iATMA is as follows. Assume that we partitioned the component types into *K* groups according to their weight values, with group-1 being the lightest and group *K* being the heaviest. Then the following equations can be used to calculate SC_i.

$$SC_K = \min\{N, N_K + NPZ\} \tag{III.99}$$

$$SC_i = \max\left\{0, \min\left\{N_i, \sum_{j=1}^{i} N_j - NPZ\right\}\right\} \qquad \text{for } 1 \le i \le K-1, \tag{III.100}$$

where *N* denotes the total number of components to be populated, N_i represents the number of components to be placed in group *i* and *NPZ* stands for the number of heads that are in no-pickup zone with $NPZ \ge 0$. It is worth to note that the formulation has no recurrence relation and it is a general formula which works for any value of K for $1 \leq K \leq N$. For the special case when $K=1$, i.e. there is only a unique weight category, we consider this unique group as the heaviest and apply Equation (III.99). Also, for $i=1$, Equation (III.100) can be reduced to

$$SC_1=\max\{0,\min\{N_1,N_1-NPZ\}\} \qquad \text{(III.101)}$$

which is equal to

$$SC_1=\max\{0,N_1-NPZ\} \text{ since } NPZ \geq 0. \qquad \text{(III.102)}$$

The performance gain obtained by applying iATMA on the considered machine will be discussed in the Results chapter.

III.3.1.1. Calculating the number of components placed in each speed category for ATMA

Recall that the secondary problem introduced previously looks for answer of the following question: Of the 80 components in group-1 which 50, 15, 10 and 5 should be selected to be placed when turret time is 0.20, 0.23, 0.33 and 0.40 s, respectively. So, the objective is to calculate the relevant M_k values.

Calculating the number of components placed in each speed category for each group may seem easy for the above example but a general calculation scheme is necessary because total assembly cost is based on it.

Firstly, SC_i values are obtained by using relevant equations given in (Duman, 2007a). Then, a rule similar to the well-known Northwest corner rule is applied. But in our case, the values in the columns add up to the SC_i values and the values in the rows must add up to N_i values. Then, the cells are filled up by moving to east (=right) as much as possible and then to the south (=down). In the rare case, we may encounter the degenerate case in which the working cell happens to meet the components exactly on both the column and the row; the algorithm will continue simply by moving east, with a zero value assigned to the new working cell. A numeric example is given below.

Assume that 100 components where N_1=82, N_2=6, N_3=6 and N_4=6 are to be populated on a PCB with n_1=40, n_2=3, n_3=3 and n_4=3. Applying the SC_i formulas, SC_1=56, SC_2=9, SC_3=9 and SC_4=26. Then, by applying the modified Northwest corner rule, values in Table III.1 can be obtained easily.

Table III.1—Example of obtaining number of components placed in each speed category for each group when ATMA is used (N=100).

ATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 82	56	9	9	8
Group-2	Out of 6	-	-	-	6
Group-3	Out of 6	-	-	-	6
Group-4	Out of 6	-	-	-	6
Total		56	9	9	26

As another example, assume that 400 components where N_1=300, N_2=40, N_3=40 and N_4=20 are to be populated on a PCB with n_1=40, n_2=3, n_3=3 and n_4=3. Applying the SC_i formulas, SC_1=274, SC_2=43, SC_3=43 and SC_4=40. Then, by applying the modified Northwest corner rule, the following table can be obtained (Table III.2).

Table III.2—Example of obtaining number of components placed in each speed category for each group when ATMA is used (N=400).

ATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 300	274	26	-	-
Group-2	Out of 40	-	17	23	-
Group-3	Out of 40	-	-	20	20
Group-4	Out of 20	-	-	-	20
Total		274	43	43	40

III.3.1.2. Calculating the number of components placed in each speed category for iATMA

Obtaining number of components placed in each speed category for each group in iATMA is derived in a different way. Firstly, SC_i values are calculated but one should note that when using iATMA, different formulas are used for calculating SC_i values. In this matrix, only five cells are filled in any case. The northeast cell is filled with 20 (number of heads in the no-pickup zone), the first diagonal element is N_1-20, and the rest on the diagonal are corresponding N_i values.

Table III.3 and Table III.4 illustrate the results when iATMA is applied to the above examples.

Table III.3—Example of obtaining number of components placed in each speed category for each group when iATMA is used (N=100).

iATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 82	62	-	-	20
Group-2	Out of 6	-	6	-	-
Group-3	Out of 6	-	-	6	-
Group-4	Out of 6	-	-	-	6
Total		62	6	6	26

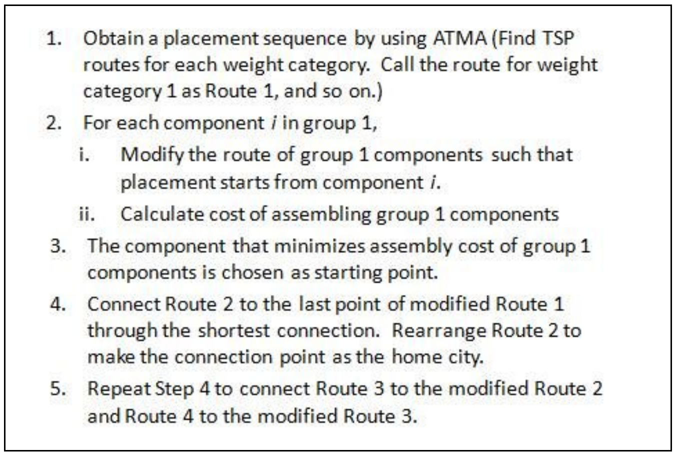
Table III.4—Example of obtaining number of components placed in each speed category for each group when iATMA is used (N=400).

iATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 300	280	-	-	20
Group-2	Out of 40	-	40	-	-
Group-3	Out of 40	-	-	40	-
Group-4	Out of 20	-	-	-	20
Total		280	40	40	40

III.3.2. Adjust First Point Procedure

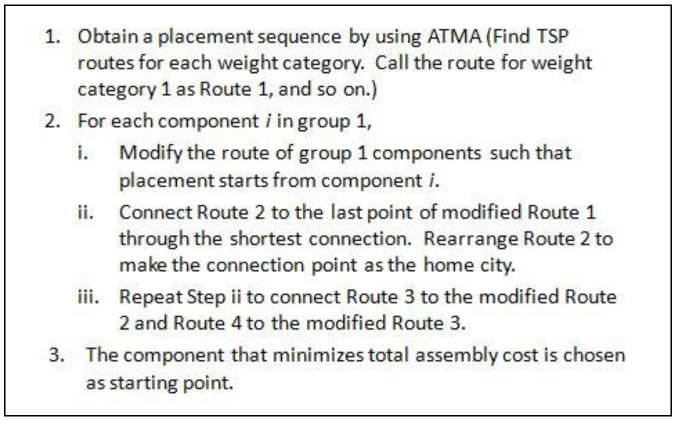
For minimizing the assembly time of PCBs, another approach we developed is based on trying different first (starting) points for the tour. Of course, this is applicable to the cases in which the preference of first point of the tour does not matter. In this technique, a TSP tour is obtained by using Convex-Hull and Or-Opt algorithms. Then, for each point *i*, total cost of assembly is recalculated as if assembly starts with point *i*. The first point which minimizes assembly cost is selected as the actual first point for the assembly process.

Adjust First Point Procedure (AFPP) and Adjust First Point according to Total Cost Procedure (AFPTCP) are developed in this understanding. In AFPP, placement cost of only group-1 components is used (Figure III.2). At that point, recall that placement cost of group-1 components may change with different starting points because determining the placement order among group-1 components turns out to be the TDTSP. However, a more realistic cost calculation should be based on placement cost of all components and AFPTCP is designed in this way (Figure III.3). There are some details that should be cared when designing AFPTCP. Remember that after the tour of group-1 components is completed, the assembly continues with group-2, group-3 and group-4 components successively. And in ATMA and iATMA first point for each successive tour is selected as the one closest to last point of precedent tour. Altering first point of group-1 results in changing the last point for group-1. Then accordingly, tours of group-2, group-3 and group-4 are modified and total cost of assembly is calculated in AFPTCP.



1. Obtain a placement sequence by using ATMA (Find TSP routes for each weight category. Call the route for weight category 1 as Route 1, and so on.)
2. For each component *i* in group 1,
   i. Modify the route of group 1 components such that placement starts from component *i*.
   ii. Calculate cost of assembling group 1 components
3. The component that minimizes assembly cost of group 1 components is chosen as starting point.
4. Connect Route 2 to the last point of modified Route 1 through the shortest connection. Rearrange Route 2 to make the connection point as the home city.
5. Repeat Step 4 to connect Route 3 to the modified Route 2 and Route 4 to the modified Route 3.

Figure III.2—Adjust First Point Procedure (AFPP)



1. Obtain a placement sequence by using ATMA (Find TSP routes for each weight category. Call the route for weight category 1 as Route 1, and so on.)
2. For each component *i* in group 1,
   i. Modify the route of group 1 components such that placement starts from component *i*.
   ii. Connect Route 2 to the last point of modified Route 1 through the shortest connection. Rearrange Route 2 to make the connection point as the home city.
   iii. Repeat Step ii to connect Route 3 to the modified Route 2 and Route 4 to the modified Route 3.
3. The component that minimizes total assembly cost is chosen as starting point.

Figure III.3—Adjust First Point according to Total Cost Procedure (AFPTCP)

We expect AFPTCP to bring improvements due to two reasons. First, AFPTCP looks for improvement opportunities that minimize total assembly cost in cases where better (closer) connection points between groups exist. Secondly, starting the tour from a different point may cause more number of placements to be done within turret time. This topic is expanded in the Results chapter with examples. Note that AFPP or AFPTCP brings no change in cost calculation when considered under classical TSP constraints. As expected, AFPTCP gives better assembly cost than AFPP.

III.3.3. Postpone Deviant

There are few studies in literature that deal with optimizing the operations of chip mounter and chip shooter placement machines by taking into consideration the turret speed changes occurring when components of different sizes are carried (Duman, 2007a, Chyu and Chang, 2008, Ellis et al., 2001). The general idea in these studies groups components according to their turret speed values, builds a TSP tour for each group and places these groups of components from lightest to heaviest or vice versa. However, in some cases this approach may produce inefficient placement sequences. Consider the following examples in which the placement sequence is built already (e.g. by ATMA); while placing a group of components, there may be some components whose placement location may be deviant from their group. A component in a placement sequence is called as deviant when either travel to or from that location exceeds turret time. Placing these components among another group of components may bring an improvement if its placement is completed within turret time. As another example, consider a component whose removal from the sequence does not increase the total assembly cost, that is, direct travel from its preceding neighbor to its succeeding neighbor is completed within turret time. Placing such a component among another group of components may reduce total assembly time by connecting two deviant components. It should be noted that placing lighter components among heavier ones may bring improvements while placing heavier ones among lighter components definitely brings inferiority in terms of total assembly time. This is because of the fact that carrying a heavier component among lighter ones increases turret time and also placement time for a number of succeeding and also preceding lighter components. Hence, starting from lightest group of components, lighter components should be tried to be inserted among heavier components. Postpone Deviant (PD) heuristic is given in Figure III.4.

```
1. Sort the component groups from lightest to heaviest
2. For each group m
   • For each component i in group m
      • For each group n heavier than m
         • For each component j in group n
            • Temporarily insert i after j and if total
              assembly cost is decreases make this insertion
              permanent
```

Figure III.4—Postpone Deviant heuristic

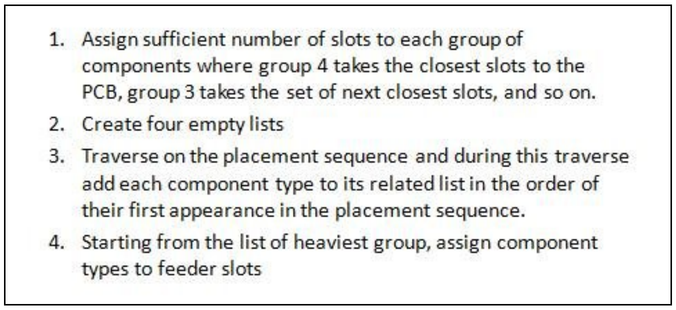
An extended version of the PD (EPD) heuristic can be defined as follows (Figure III.5).

```
Do
   1. assemblyTime = total assembly time of current
      placement sequence
   2. Call PD
   3. newAssemblyTime = total assembly time of current
      placement sequence
While newAssemblyTime ≠ assemblyTime
```

Figure III.5—Extended PD heuristic

The EPD calls the PD heuristic until no improvements are obtained in assembly time. It is expected that the EPD requires more time but gives better assembly time values. Extended analyses are given in Results chapter.

PD and EPD heuristics can be applied to chip mounter and chip shooter machines. However, when applying these heuristics to chip mounter machine, some modifications must be done to FAP. Recall that FAP is a procedure that makes the feeder configuration and is applied after the placement sequence is built. FAP arranges the feeder slots by the assumption that components of each weight category are placed successively and FAP is effective only when this assumption is satisfied. However, PD (or EPD) may mix the component groups. Hence, FAP must be modified to perform well under these new conditions. We propose modified FAP heuristic as follows.



1. Assign sufficient number of slots to each group of components where group 4 takes the closest slots to the PCB, group 3 takes the set of next closest slots, and so on.
2. Create four empty lists
3. Traverse on the placement sequence and during this traverse add each component type to its related list in the order of their first appearance in the placement sequence.
4. Starting from the list of heaviest group, assign component types to feeder slots

Figure III.6—Modified FAP

III.3.4. Group Insertion and Individual Insertion Heuristics

In order to obtain further improvements in terms of assembly time, new approaches can be proposed by taking advantage of inherent design of the analyzed PCBs. Figure III.7 depicts a typical PCB produced in the industry. Recall that, in ATMA route for each group is built firstly, and then first point for group-1 is selected for starting the assembly process (we preferred to select the one at southwest). Then, route 2 is connected to the last point of route 1 through shortest connection and route 2 is rearranged so that first point of route 2 is the connection point. This adjustment is repeated for the route pairs (route 3, route 2) and (route 4, route 3), successively. After these adjustments, last point of route 4 may stand in a position that is far away from first point of route 1, as illustrated in Figure III.7. When a continuous production is considered, the cost of this connection gains importance and its reduction should be sought.

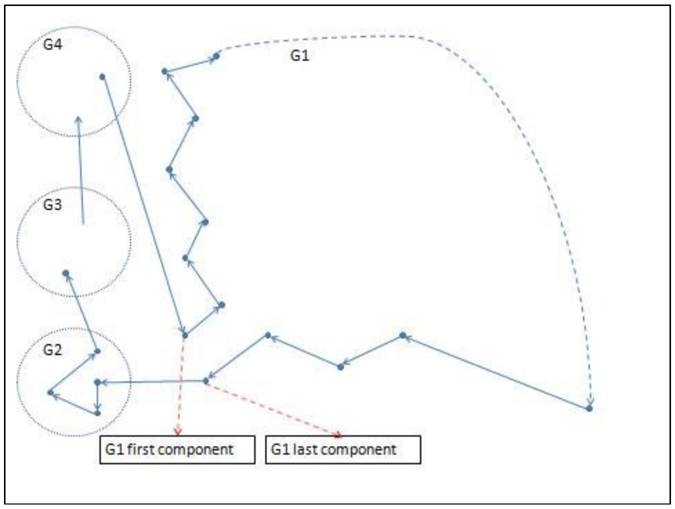


Figure III.7—Example depicting solution of ATMA

To cope with this deficiency, we propose making modifications to the solution obtained by ATMA. We alter the first point of group-1 as the one closest to the last point of group-4. This must be done by preserving the last point of group-1, because other routes are already built according to last point of group-1. However, in this case, some of the points (points between last point of group-4 and new first point of group-1) are excluded from the tour and this situation raises the question of how to deal with these excluded components (see Figure III.8). These components somehow must be inserted into the placement sequence.

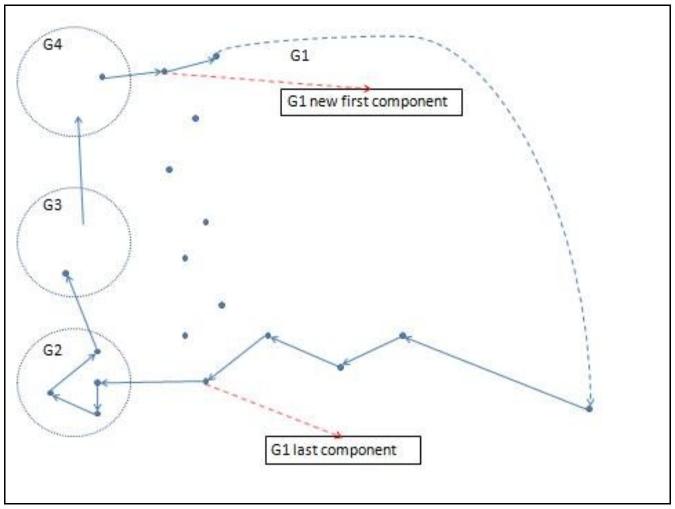


Figure III.8—Excluded vertices if first point is altered.

For inserting these components into the placement sequence, individual insertion (II) heuristic can be used (Figure III.9).

```
1. Put all excluded points in a set EP
2. While EP is not empty
    i.   Choose the point k from EP and the edge (i,j) from tour
         such that cost of inserting k between i and j is minimum
    ii.  Insert point k in tour between i and j
    iii. Remove k from EP
```

Figure III.9—Individual Insertion Heuristic (II)

II method tries to insert components one-by-one. Within the excluded points (EP) set, we find the vertex k and edge *(i,j)* pair which gives the minimum cost and insert vertex k between edge *(i,j)*. This process is repeated until EP is not empty. Or discusses three different cost measures for inserting points into a tour (Or, 1976).

These are difference, ratio and multiplication (difference times ratio). We tried all of these measures, but we followed up the difference approach since it produced better results.

A possible outcome of inserting excluded points into the tour is illustrated in Figure III.10.

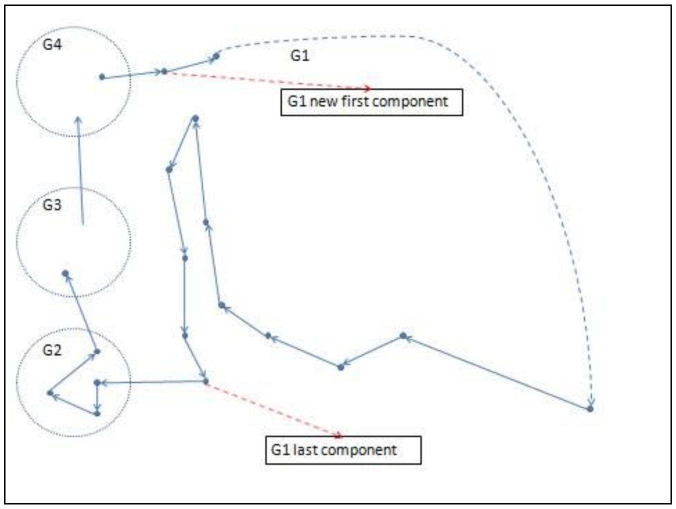


Figure III.10—A possible outcome of inserting excluded points

Another method for inserting these excluded points can be defined as follows. The points starting from the point closest to the last point of group-1, to the point closest to new first point of group-1 (points between *a* and *b* in Figure III.11) are separated from the tour by preserving the order determined by Convex-Hull Or-Opt. Then insert this group of points in the tour where they give the minimum total cost (total assembly time). In each insertion trial, both normal and the reverse order of the group are considered. This approach for inserting the excluded points is called Group Insertion (GI).

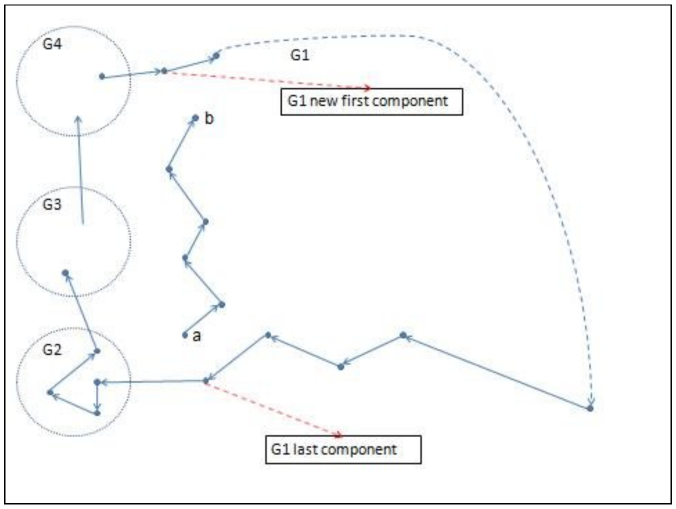


Figure III.11—Connecting excluded vertices

One should keep in mind that the above discussions are valid for ATMA. A similar discussion can be made for iATMA by taking into consideration the inverse flow of placement sequence. As observed in Figure III.12, the distance between last point of group-1 and first point of group-4 is very great and counts for an increase in assembly time when continuous production is considered. Hence, altering last point of group-1 should be investigated in this case.

Finding the new last point for group-1 components is very easy. If we simply connect new last component of group-1 to first component of group-4, then we are faced with a situation as depicted in Figure III.13.

In this figure we see that the excluded points from the tour are the ones that are on the right of the PCB. However, due to the inherent nature of analyzed PCBs, in most cases these excluded points are more than the points which are included in the tour. In this case, we will insert more number of points into a tour with less number of points. Thus, it seems more rational to exchange the included points with excluded ones (Figure III.14).

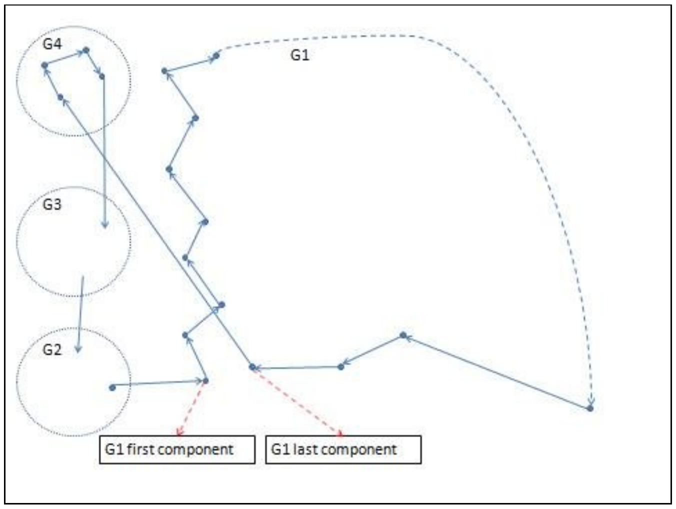


Figure III.12—Example depicting solution of iATMA

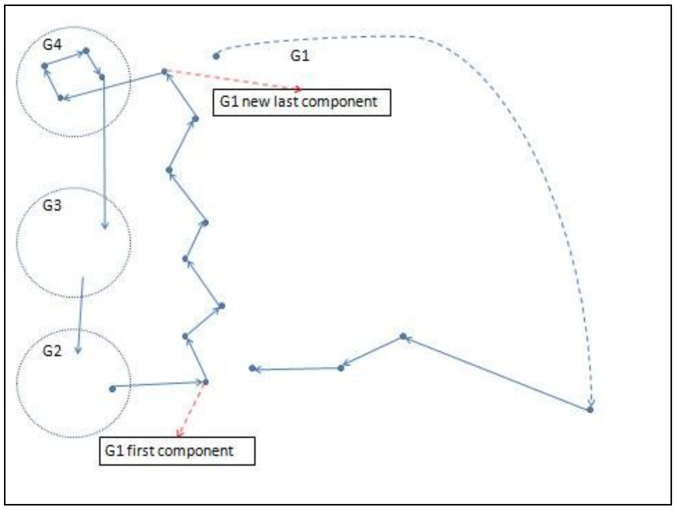


Figure III.13—Connecting last point of group-1

Another important detail in this design is the direction of the tour. Note that, it is required to reverse the direction of the tour from first component of group-1 up to new last component of group-1.

After these adjustments, individual insertion and group insertion methods are applied after iATMA.

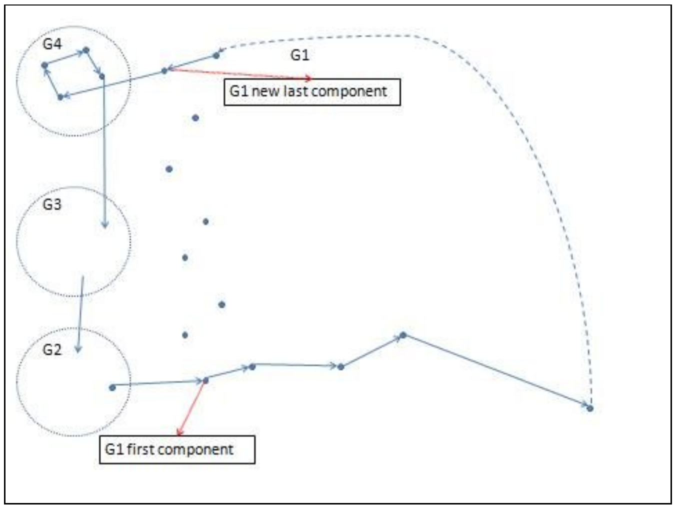


Figure III.14—Reversing the tour of group-1

III.3.5. Record-to-Record with Local Exchange Moves (RRTLEM)

Record-to-Record Travel with Local Exchange Moves (RRTLEM) is a local search method that combines Record-to-Record Travel (RRT) and local exchange moves. As pointed out in previous chapter the main objective of local search methods is to improve a given solution by searching the solution space locally.

RRT is developed as a deterministic variant of Simulated Annealing (SA) and it is shown that the quality of the computational results obtained so far by RRT is better than SA (Dueck 1993). In a recent study, (Li et al., 2007) used RRT in solving HVRP.

RRT starts by obtaining an initial solution, *GenerateInitialSolution()*. In PCB assembly context, a solution refers to a placement sequence and feeder arrangement. *GenerateInitialSolution()* may be either be a constructive method or a local search method. In our case, ATMA (or iATMA) gives the initial solution. Generated initial solution is assigned to *s*, which stores current solution. Variable *bs* keeps track of best solution found within the algorithm and initially *bs* is set to *s*. Function *f(s)* returns the cost of a given solution, *s*. In our case, *f(s)* calculates the total assembly cost for a given placement sequence. *Record* is defined as the cost of best solution observed so far. *Deviation* is defined as a predefined percentage (*p*) of *Record*. As it was pointed out, local search methods require searching the solution space locally. *N(s)* specifies the set of all other solutions that are neighbors of *s* and the function *PickNextNeighbor()* retrieves the next neighbor solution, *s'*, from *N(s)* by a predefined rule or order. RRT is deterministic because, a neighbor solution *s'* replaces current solution only if its cost is less than *Record+Deviation*. The pseudo code of RRT is given in Figure III.15.

```
s := GenerateInitialSolution()
bs := s
Record := f(bs)
Deviation := Record x p
While termination conditions not met
    s' := PickNextNeighbor(N(s))
    If f(s') < Record + Deviation
        s := s'
        If f(s') < Record
            bs := s'
            Record := f(s')
            Deviation := Record x p
        Endif
    Endif
Endwhile
```

Figure III.15—Record to Record Travel (RRT)

Obtaining new solutions from current solution requires a local search. In TSP variants, a general local search method is to apply exchanges of edges or nodes (Waters, 1987; Croes, 1958). For exchanging two nodes, two alternatives exist; we can either insert a node into a different place on the route, called 1-0 Exchange move, or we can exchange two nodes, called 1-1 Exchange. Exchanging two edges is called 2-Opt move. They are explicitly depicted in Figure III.16-Figure III.18.

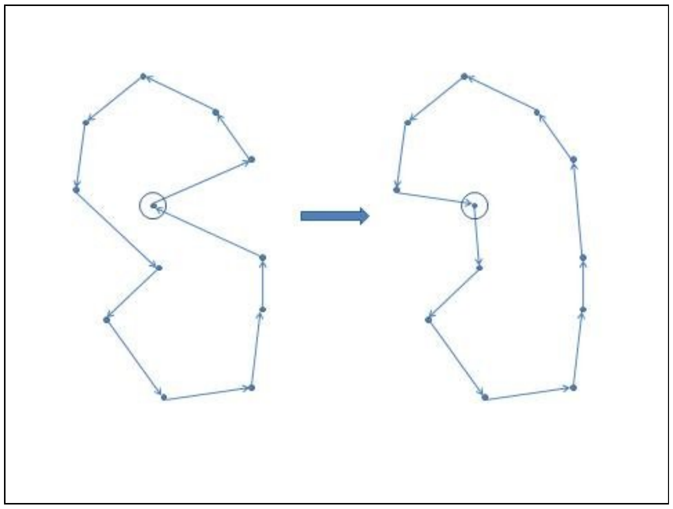


Figure III.16—1-0 Exchange move

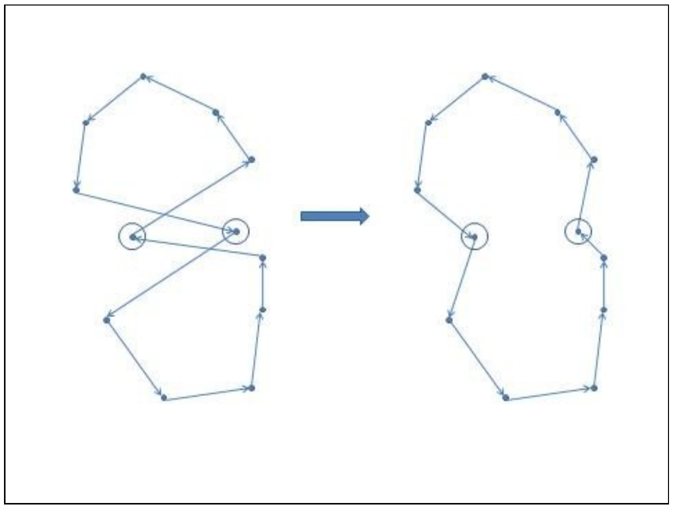


Figure III.17—1-1 Exchange move

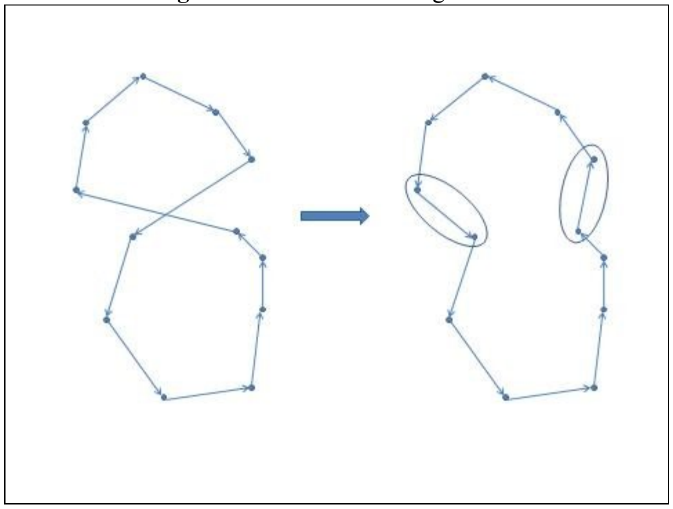
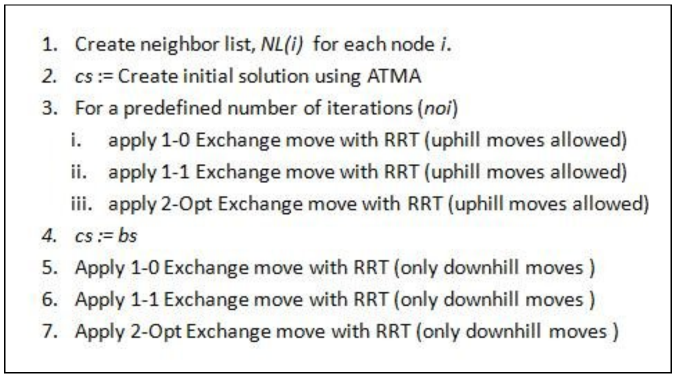


Figure III.18—2-Opt move

We developed an improvement algorithm that combines RRT with local search moves (RRTLEM). It is a general framework consists of simple iterative statements (Figure III.19). RRT concept is embedded in local search moves (Figure III.20-Figure III.22). Recall that in the problem that we explore, there are four groups of components which are placed sequentially. ATMA creates these four routes, and then combines them. The actual point that we try to improve is the sequencing of group-1 components. Thus RRT is applied only within group-1 components.



```
1. Create neighbor list, NL(i) for each node i.
2. cs := Create initial solution using ATMA
3. For a predefined number of iterations (noi)
    i.   apply 1-0 Exchange move with RRT (uphill moves allowed)
    ii.  apply 1-1 Exchange move with RRT (uphill moves allowed)
    iii. apply 2-Opt Exchange move with RRT (uphill moves allowed)
4. cs := bs
5. Apply 1-0 Exchange move with RRT (only downhill moves )
6. Apply 1-1 Exchange move with RRT (only downhill moves )
7. Apply 2-Opt Exchange move with RRT (only downhill moves )
```

Figure III.19—RRT with Local Exchange Moves (RRTLEM)

Neighbor list is an important concept that should be explained. During local searches, it is more rational to try exchanges between nodes or edges that are away from each other at most a reasonable amount of distance. For example, for exchanging two nodes, it is vainly to exchange the nodes at the most west and the most east. To avoid from these unprofitable moves, we build neighbor lists for each node. For chip mounter machines, *NL(i)* is a set that consists of the nodes that are at most 48 mm away from node *i* (using Chebyshev metric, thus it forms a square not a circle). 48 mm is the distance that the board carrier can move within turret time when carrying the components of heaviest type (0.4x120).

As the initial solution, we will use the output of ATMA and then try to improve it by applying local search moves for a predefined number of iterations. During the search process, current solution (*cs*) is the placement sequence from which new solutions (*s'*) are obtained and best solution (*bs*) is the route whose cost is the best among the solutions created. Cost of a solution *s* is denoted by *f(s)* and it is the total assembly time of the PCB using the placement sequence stored in *s*. When uphill moves are allowed for a local search move, *cs* can be assigned a new solution whose cost is worse than cost of *cs* but better than cost of best solution plus deviation. Deviation is a predefined percentage (*rate*) of cost of best solution, and whenever a new best solution is found deviation is also updated. Uphill moves create the opportunity for *cs* to escape from trapping local minima. After executing iterative statements, starting from the *bs*, we apply each local search move once more.

The idea is inspired from the work of (Li et al., 2007) but important modifications are added that could improve the performance. Note that in our algorithm, there are two parameters that must be fine tuned in order to obtain best performance. These are the number of iterations, *noi*, and percentage value, *rate*, which is used to calculate the deviation. They are analyzed in detail in Results section. Since it is an improvement heuristic, it can be applied after any other heuristic.

```
For each node $i$ in the current solution $cs$
    $savings := -\infty$
    For each edge $j$ whose one end is in $NL(i)$
        Obtain $s'$ by inserting node $i$ between edge $j$
        If $f(s') < f(cs)$
            $cs := s'$
            If $f(s') \leq f(bs)$
                $bs := s'$, $dev := f(bs) \times rate$
            Endif
            Continue with the next node
        Endif
        If $f(s') \geq f(s)$ and $f(s) - f(s') \geq savings$
            Store $j$ as maximum saving edge ($mes := j$)
        Endif
    Endfor
    If $f(s) - savings \leq f(bs) + dev$ and uphill moves allowed
        $cs :=$ the solution obtained by inserting node $i$ between $mes$
    Endif
Endfor
```

Figure III.20—1-0 Exchange move with RRT

```
For each node $i$ in the current solution cs
    $savings := -\infty$
    For each node $j$ which is in $NL(i)$
        Obtain $s'$ by exchanging the places of nodes $i$ and $j$
        If $f(s') < f(cs)$
            $cs := s'$
            If $f(s') \leq f(bs)$
                $bs := s'$, $dev := f(bs) \times rate$
            Endif
            Continue with the next node
        Endif
        If $f(s') \geq f(s)$ and $f(s) - f(s') \geq savings$
            Store j as maximum saving node ($msn := j$)
        Endif
    Endfor
    If $f(s) - savings \leq f(bs) + dev$ and uphill moves allowed
        $cs :=$ the solution obtained by exchanging nodes $i$ and $msn$
    Endif
Endfor
```

Figure III.21—1-1 Exchange move with RRT

```
For each edge $i$ in the current solution $cs$
    $savings := -\infty$
    For each edge $j$ whose one end is in $NL$ of both ends of $i$
        Obtain $s'$ by reconnecting the edges $i$ and $j$ in the alternative way
        If $f(s') < f(cs)$
                $cs := s'$
                If $f(s') \leq f(bs)$
                        $bs := s'$, $dev := f(bs) \times rate$
                Endif
                Continue with the next node
        Endif
        If $f(s') \geq f(s)$ and $f(s) - f(s') \geq savings$
                Store $j$ as maximum saving edge ($mse := j$)
        Endif
    Endfor
    If $f(s) - savings \leq f(bs) + dev$ and uphill moves allowed
        $cs :=$ the solution obtained by reconnecting the edges $i$ and $mse$ in
                the alternative way
    Endif
Endfor
```

Figure III.22—2-Opt Exchange move with RRT

III.3.6. Pair-wise Exchange Procedure (PEP)

In order to improve the algorithms, we designed a local random search procedure. The procedure starts with a given solution and applies a number of random pair-wise (1-1) exchanges. Therefore we call it pair-wise exchange procedure (PEP). More specifically, it can be defined as follows.

Randomly determine two points (components) within group-1 (choosing among all groups will definitely worsen the total cost because; the basic idea of our algorithms is to place groups of components successively). If the new cost stays the same or decreases by exchanging these two points, perform the exchange. Continue this procedure until a predetermined number of exchange trials are made.

PEP and RRTLEM are improvement heuristics that can be applied after any solution and if optimum solution is not reached it is likely that these local search heuristics may improve the solution. From the implementer's point of view, PEP is a straightforward algorithm to implement while RRTLEM requires use of complicated data structures and much more care.

CHAPTER IV

OPTIMIZING THE OPERATIONS OF CHIP SHOOTER MACHINES

This chapter gives the formulation of problems emerging from the operations of chip shooter machines which takes into consideration the different turret time values for different weight categories of components. In section one, the formulations of the placement sequencing problem, feeder configuration problem and the assembly time minimization problem (the integrated problem) are given. Then in section two, a discussion on a general methodology for optimizing chip shooter machines with different specifications is provided.

IV.1. PROBLEM FORMULATION OF CHIP SHOOTER PLACEMENT MACHINES

This section first gives the definitions and the notation to be used in the following formulations. Afterwards, the nonlinear integer programming formulation of the placement sequencing and feeder configuration problems will be given. Then, the assembly time minimization problem (which is comprised of these two problems) for the chip shooter machine and its nonlinear integer programming formulation will be given. We should remind that all of the following formulations assume that the placement machine has various turret time values for different component types, which is the real case as we pointed out in Chapter two.

IV.1.1. Definitions and Notation

Below, the notation used for modeling the operations of chip shooter machines is given. Some of these definitions are given in previous chapters, however, for the sake of compactness they are also provided here.

H: number of heads on the turret

N: total number of components to be placed,

n: number of component types, number of feeder slots (one component type is assigned only one slot)

K: number of weight categories

N_k: number of components in each weight category *k*, *k=1,2,…,K*.

n_k: number of component types in each weight category *k*, *k=1,2,…,K*.

d: number of heads between pickup and placement heads (*d=H/2-1*)

ct_{it}: component type matrix (*N* x *n*) indicating the component type for each component

$$ct_{it} = \begin{cases} 1 & \text{if component } i \text{ is of component type } t \\ 0 & \text{otherwise} \end{cases}$$

g_{tk}: group (weight category) matrix (*n* x *K*) indicating the group for each component type

$$g_{tk} = \begin{cases} 1 & \text{if component type } t \text{ is in group } k \\ 0 & \text{otherwise} \end{cases}$$

tt_k : turret rotation time for each weight category *k*, (When *k=1*, the turret rotation speed is the maximum, that is, turret rotation time is minimum) (*tt* is used if weight category among components is disregarded)

CF1(x, y): time required for the board carrier to move from point *x* to point *y*.

CF4(r, s): time required for the feeder carriage to travel from feeder *r* to feeder *s*.

p: is the placement order or placement position.

x_{ip}: denote the assignment of node *i* to position *p*.

$$x_{ip} = \begin{cases} 1, & \text{if node i is visited in p}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

y_{tr}: state the status of assigning component type *t* to feeder *r*.

$$y_{tr} = \begin{cases} 1, & \text{if component type } t \text{ is stored in feeder } r \\ 0, & \text{otherwise} \end{cases}$$

w_{ijp}: is used for expressing the travel from node *i* visited in $(p\text{-}1)^{th}$ position to node *j* visited in p^{th} position.

$$w_{ijp} = \begin{cases} 1, & \text{if node j is visited in p}^{\text{th}} \text{ position after node i was visited in (p - 1)}^{\text{th}} \text{ position} \\ 0, & \text{otherwise} \end{cases}$$

In this notation, the definition of the cost function *CF1(x, y)* is given in (III.1). On the other hand, *CF4(r, s)* gives time required for the feeder carriage to travel from feeder *r* to feeder *s*. We assume a linear speed for the feeder carriage (no acceleration or deceleration) and it can be stated by the following equation.

$$CF4(r,s) = |r - s| ft \tag{IV.1}$$

where *ft* is a predefined time value for the feeder magazine to move one slot.

As it is pointed out before, the operations of chip shooter machines result in two interdependent optimization problems. These are the placement sequencing and the feeder configuration problems. Formulation of both of these problems and besides the integrated problem will be given in the following subsections.

IV.1.2. Placement Sequencing Problem

Given a feeder configuration, the objective of the placement sequencing problem is to find a placement sequence of the components so that the assembly process is completed in the shortest time possible. We start the formulation by defining the cost of travel from component *i* to component *j* when component *j* is assembled in p^{th} position (C_{ijp}).

$$C_{ijp} = w_{ijp} \max \left(\begin{array}{l} CF1(i,j), \\ \max\limits_{m=0}^{d} \left(\sum\limits_{l=1}^{N} \sum\limits_{t=1}^{n} \sum\limits_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+m} \right), \\ \sum\limits_{u}^{N} \sum\limits_{v}^{N} CF4(r,s) x_{u,p+d} x_{v,p+d+1} \end{array} \right) \tag{IV.2}$$

In this formulation, first term in outer max function is the time for board carrier to align the PCB under the placement head of the turret. The second term finds the maximum of the turret time values of the following *d+1* components (including component *j*) that are carried by the heads. This is stated by a maximum function and within the maximum function the turret time values for the corresponding components to be placed in p^{th}, $(p+1)^{st}$, …,$(p+d)^{th}$ positions are calculated. While the turret is rotating one step, the feeder magazine must move to get ready for the pickup of the component to be placed in the $(p+d+1)^{th}$ position. Hence, the third term finds the time for the feeder magazine to move from slot r to slot s (given the feeder configuration, the corresponding slots of types of the *(p+d)*th and the *(p+d+1)*th components can be detected easily and it is assumed that they are stored in feeders r and s, respectively).

It is important to note a detail about indices used in formulations. In mass production environments, the components arrive at the placement location continuously and as soon as the assembly of a board is completed the components on the heads are placed on the next board. So, travelling along the path restarts after a previous tour ends. Hence, in the formulations, if an index exceeds its upper limit, it should be considered as if the counting of indices restart from one. For example, if we are talking about $(p\text{-}1)^{th}$ placement when p is 1 and N is 100, then $(p\text{-}1)^{th}$ placement means 100th placement on the previous board.

After removing the nonlinearity in C_{ijp} by introducing D_{ijp} as the dominating factor in C_{ijp} definition, we obtain the following nonlinear integer programming formulation of the problem.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \tag{IV.3}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \tag{IV.4}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{lp} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{IV.5}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+1} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{IV.6}$$

$$\vdots$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+d} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{IV.7}$$

$$D_{ijp} - w_{ijp} \sum_{u}^{N} \sum_{v}^{N} CF4(r,s) x_{u,p+d} x_{v,p+d+1} \geq 0 \qquad i,j,p = 1,\ldots,N \tag{IV.8}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\dots,N, \qquad \text{(IV.9)}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\dots,N, \qquad \text{(IV.10)}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\dots,N, \qquad \text{(IV.11)}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\dots,N, \qquad \text{(IV.12)}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\dots,N, \qquad \text{(IV.13)}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\dots,N, \qquad \text{(IV.14)}$$

We introduce several constraint sets to the problem. Constraint set (IV.4) imposes the requirement that placement time for component *j* (component *j* being placed after component *i* in p^{th} position) cannot be smaller than the travel time from *i* to *j*. Constraint sets (IV.5-IV.7) (representing *d+1* different constraint sets) impose that placement time for component *j* cannot be smaller than the turret time values associated with the incoming components for placement. Constraint (IV.8) guarantees that placement time for component *j* cannot be smaller than the movement of feeder magazine. Constraint (IV.9) guarantees that each component is assigned to a position and constraint (IV.10) guarantees that each position is occupied by only one node. There is no need for constraints that are used to guarantee subtour elimination, because placing each node in exactly one position is achieved by these constraints. Constraint (IV.11) states that if component *j* is assigned to p^{th} position, then exactly one component precedes *j* in the travel sequence. Similarly (IV.12) states that if component *i* is assigned to $(p\text{-}1)^{\text{th}}$ position then exactly one component procedes it in the travel sequence. Constraints (IV.13) and (IV.14) are the binary constraints for decision variables x_{ip} and w_{ijp}.

In the next subsection, formulation of the feeder configuration problem will be introduced.

IV.1.3. Feeder Configuration Problem

Given a placement sequence, the objective of the feeder configuration problem is to find a configuration of the component types to slots so that the assembly process is completed in the shortest time possible. For a given placement sequence, the feeder configuration problem can be considered as a QAP (Duman and Or, 2007). However, in the formulation of cost function we should take into account the turret time, traveling time of the board carrier and the movement time of the feeder carriage. Thus, we preferred to formulate the problem by considering the objective function as minimizing the total assembly time of the components. In this way, the formulation of the combined problem will be easier.

Then, cost of placement of component *j* when it is assembled in p^{th} position and component *i* is assembled in $(p\text{-}1)^{\text{th}}$ position, C_p, can be calculated as follows:

$$C_p = \max\left(\begin{array}{l} CF1(i,j), \\ \max\limits_{m=p}^{p+d}\left(\sum\limits_{t=1}^{n}\sum\limits_{k=1}^{K} tt_k g_{tk} ct_{mt} \right), \\ \sum\limits_{t_1}^{n}\sum\limits_{t_2}^{n}\sum\limits_{r}^{n}\sum\limits_{s}^{n} CF4(r,s) y_{t_1,r} ct_{u,t_1} y_{t_2,s} ct_{v,t_2} \end{array} \right) \qquad \text{(IV.15)}$$

In this formulation, *u* and *v* denote the components planned to be placed in $(p+d)^{th}$ and $(p+d+1)^{th}$ positions, respectively. Also, ct_{mt} gives the component type of the component placed in m^{th} position. In the formulation, first term in outer max function is the time for board carrier to align the PCB under the placement head of the turret. The second term finds the maximum of the turret time values of the following $d+1$ components to be placed in p^{th}, $(p+1)^{\text{st}}$, …,$(p+d)^{\text{th}}$ positions. Note that, since the placement order of each component is known, former two terms can be directly obtained. While the turret is rotating one step, the feeder magazine must move to get ready for the pickup of the component to be placed in the $(p+d+1)^{th}$ position. Hence, the third term finds the time for the feeder magazine to move from slot *r* to slot *s* by incorporating decision variables y_{tr}.

After removing the nonlinearity in C_p by firstly introducing D_p as the dominating factor in C_p definition, we obtain the following nonlinear integer programming formulation of the problem.

$$\min \sum_{p=1}^{N} D_p \qquad \text{(IV.16)}$$

s.t.

$$D_p - CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.17)}$$

$$D_p - \sum_{t=1}^{n}\sum_{k=1}^{K} tt_k g_{tk} ct_{pt} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.18)}$$

$$D_p - \sum_{t=1}^{n}\sum_{k=1}^{K} tt_k g_{tk} ct_{p+1,t} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.19)}$$

$$\vdots$$

$$D_p - \sum_{t=1}^{n}\sum_{k=1}^{K} tt_k g_{tk} ct_{p+d,t} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.20)}$$

$$D_p - \sum_{t_1}^{n}\sum_{t_2}^{n}\sum_{r}^{n}\sum_{s}^{n} CF4(r,s) y_{t_1,r} ct_{u,t_1} y_{t_2,s} ct_{vt_2} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.21)}$$

$$\sum_{r=1}^{N} y_{tr} = 1, \qquad t = 1,\ldots,N, \qquad \text{(IV.22)}$$

$$\sum_{t=1}^{N} y_{tr} = 1, \qquad r = 1,\ldots,N, \qquad \text{(IV.23)}$$

$$y_{tr} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \qquad \text{(IV.24)}$$

Constraint set (IV.17) imposes the requirement that placement time for component *j* (component *j* being placed after component *i* in *p*th position) cannot be smaller than the travel time from *i* to *j*. Constraint sets (IV.18-IV.20) (representing *d+1* different constraint sets) impose that placement time for component *j* cannot be smaller than the turret time values associated with the transferred components. Constraint (IV.21) guarantees that placement time for component *j* cannot be smaller than the movement of feeder magazine. Constraint (IV.22) guarantees that each component type is assigned to a feeder slot and constraint (IV.23) guarantees that each feeder slot is occupied by only one component type. Constraints (IV.24) are the binary constraints for decision variables y_{tr}.

In the next subsection, the assembly time minimization problem (integrated version of placement sequencing and feeder configuration problems) will be formulated.

IV.1.4. Assembly Time Minimization Problem

Recall that the ultimate objective of the optimization problem for chip shooter machines is to minimize the total assembly time. That is, the objective is to find a placement sequence of components and a feeder configuration so that the assembly process is completed in the shortest time possible. So the complete and integrated model can be formulated by firstly defining the cost function of assembling component *j* just after component *i* in *p*^th^ position.

$$C_{ijp} = w_{ijp} \max \begin{pmatrix} CF1(i,j), \\ \max\limits_{m=0}^{d} \left(\sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+m} \right), \\ \sum_{t_1}^{n} \sum_{t_2}^{n} \sum_{r}^{n} \sum_{s}^{n} \sum_{u}^{N} \sum_{v}^{N} CF4(r,s) x_{u,p+d} x_{v,p+d+1} y_{t_1,r} ct_{u,t_1} y_{t_2,s} ct_{v,t_2} \end{pmatrix} \qquad \text{(IV.25)}$$

After removing the nonlinearity in C_{ijp} by introducing D_{ijp} as the dominating factor in C_{ijp} definition, we obtain the following nonlinear integer programming formulation of the problem. Note that this formulation incorporates both x_{ip} and y_{tr} variables.

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{ijp} \qquad \text{(IV.26)}$$

s.t.

$$D_{ijp} - w_{ijp} CF1(i,j) \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.27)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{lp} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.28)}$$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+1} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.29)}$$

$\vdots$

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} tt_k g_{tk} ct_{lt} x_{l,p+d} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.30)}$$

$$D_{ijp} - w_{ijp} \sum_{t_1}^{n} \sum_{t_2}^{n} \sum_{r}^{n} \sum_{s}^{n} \sum_{u}^{N} \sum_{v}^{N} CF4(r,s) x_{u,p+d} x_{v,p+d+1} y_{t_1,r} ct_{u,t_1} y_{t_2,s} ct_{v,t_2} \geq 0 \qquad i,j,p = 1,\ldots,N \qquad \text{(IV.31)}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1,\ldots,N, \qquad \text{(IV.32)}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1,\ldots,N, \qquad \text{(IV.33)}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \qquad j,p = 1,\ldots,N, \qquad \text{(IV.34)}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \qquad i,p = 1,\ldots,N, \qquad \text{(IV.35)}$$

$$\sum_{r=1}^{N} y_{tr} = 1, \qquad t = 1,\ldots,N, \qquad \text{(IV.36)}$$

$$\sum_{t=1}^{N} y_{tr} = 1, \qquad r = 1,\ldots,N, \qquad \text{(IV.37)}$$

$$x_{ip} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \qquad \text{(IV.38)}$$

$$w_{ijp} \in \{0,1\}, \qquad i,j,p = 1,\ldots,N, \qquad \text{(IV.39)}$$

$$y_{tr} \in \{0,1\}, \qquad i,p = 1,\ldots,N, \qquad \text{(IV.40)}$$

What these constraints correspond to are explained in the previous subsections in detail. Therefore, there is no need to explain them here again.

IV.2. A GENERAL METHODOLOGY FOR OPTIMIZING CHIP SHOOTER MACHINES

There are several studies that analyze chip shooter machines and optimization problems emerging from their operation principles. However, all of them focus on a specific machine model and deal only with its optimization based on some assumptions made. A solution methodology developed for a specific machine may not be effective for another because different assumptions yield different optimization problems. On the other hand, a small assumption or relaxation of a constraint may initiate an opportunity that brings further optimization to the placement machine. In the literature, to the best of our knowledge, there is no study that proposes a general methodology that is applicable to any type of machine.

The solution methodology proposed for a chip shooter machine is mainly based on considering the opportunities arising from the relative speed values of the three basic mechanisms of it. Therefore, in order to develop a general methodology that is applicable for any type of chip shooter machines, firstly, we should construct a speed classification scheme for various relative speeds among the three basic mechanisms. The turret has a rotation speed and board carrier (BC) and feeder magazine (FM) has movement speeds. The dominating factor in building a model is the rotation speed of the turret, because it creates free time for other mechanisms. Thus, we will use terms slow, moderate and fast relative to the speed of turret. What do these terms correspond to numerically are as follows.

Table IV.1—Speed Classification Scheme

	Distance that BC can move freely in units of average distance between component pairs (*x*)	Number of slots that FM can move freely (*m*)
Fast	$x \geq 5$	$m \geq 5$
Moderate	$1 \leq x < 5$	$1 \leq m < 5$
Slow	$x < 1$	$m < 1$

These definitions arise nine different cases with their special constraints. These nine cases are defined below with our proposed solution heuristics.

1. Fast BC and fast FM: In this case, turret speed highly influences the total assembly time and assembly time values very close to lower bound can obtained. Both feeder configuration and placement sequencing problems can be solved to optimality, because ideal movement sequences that does not exceed the free time for both feeder carriage and board carrier could be easily found. For this integrated problem, firstly building a placement sequence and then a feeder configuration, or reversing the process should yield near optimal results. Solving these problems iteratively could result in optimum solutions.

2. Fast BC and moderate FM: In this case, moving the FM more than free time should be avoided as much as possible. Hence, the integrated problem will be dominated by the quadratic assignment problem, because feeder carriage movement times would mostly exceed rotation time. A possible strategy can be obtained by firstly dividing the component types into *n/m* number of sets and then building a placement sequence for each set.
3. Fast BC and slow FM: In this case, movement of FM should be avoided as much as possible. A placement sequence for each component type should be built and so components belonging to a type can be processed without moving the FM. In other words, placing all components of the same type consecutively so that device table only moves each time a new type of components is started to place. This method is also applied in (Kumar and Luo, 2003).
4. Moderate BC and fast FM: The movements of BC become the dominant factor for assembly time, in this case. The integrated problem turns out to be almost the TSP because between any consecutive placements, the FM is capable of picking up the next component from a different slot on time. Thus the problem can be solved by any efficient heuristic developed for TSP.
5. Moderate BC and moderate FM: The most tackling problems arise in this condition because the integrated problem includes optimization of both FM and BC movements. Hence, concurrent solution of both problems is required. Reaching the optimum becomes really hard and machine specific parameters should be analyzed very well.
6. Moderate BC and slow FM: Unnecessary movements of FM should be avoided. Best practice for minimizing assembly time seems placing all components of the same type consecutively so that device table only moves each time a new type of components is started to place.
7. Slow BC and fast FM: This time, movements of BC should be minimized as much as possible. After building placement sequence, feeder configuration can be built using this placement sequence.
8. Slow BC and moderate FM: This time, movements of BC should be minimized as much as possible by taking care of lengthy movements of FM. A possible strategy can be obtained by firstly dividing the component types should be into *n/m* number of sets and a placement sequence for each set should be built.

9. Slow BC and slow FM: In this case, the effect of turret time on assembly is minimized because it has a fast operating capability when compared with BC and FM. Hence, optimization problem requires optimization of movements of both mechanisms. Reaching the lower bound becomes almost impossible and machine specific parameters should be analyzed very well. A strategy can first build a TSP tour for the components and the feeder configuration can be built according to the flow matrix obtained from this tour.

In this study, we take into account the realistic constraint that the placement machine has various turret time values for different component types. Thus, the components should be divided into groups according to their turret time values. For each group, after determining corresponding case according to the speed category of FC and FM, the corresponding actions given above should be taken. So, the problem is divided into subproblems.

For each subproblem, if the placement sequence is to be found, a constructive heuristic and improvement heuristics can be used for obtaining an initial solution in low run time. After building all subtours for all subproblems, they must be connected by the closest connection point. After building the tour, the proposed heuristics in this study can be applied to obtain further improvements.

On the other hand, if the feeder configuration is to be found, we can firstly apply a constructive heuristics. As an example, put the component type of the starting component to most left of the feeder magazine. Then continue placing the component types on the feeder magazine according to the tour list visited in the placement sequence. Another method to build feeder configuration may be by using a flow matrix (Vittes, 1999).

Lastly, any improvement heuristics such as solving the placement sequencing or feeder configuration problems iteratively until no improvements are obtained or meta heuristics such as GA, SA can be applied for further improvements.

CHAPTER V

RESULTS AND DISCUSSION

In this chapter, we give an extensive analysis and comparison study of the techniques described in the previous chapters which are applied on chip mounter machines.

V.1. RANDOMLY GENERATED BENCHMARK DATA

A data generator that produces random printed circuit board assembly data is implemented. The position, type of the component and which category it belongs to is randomly created. The methodology for this generator is the same as the one used by (Duman, 2007a). The total number of components to be placed (*N*) is prespecified and it is given to the data generator. We studied with four different *N* values, 100, 200, 300 and 400. For *N*=100, the number of component types in groups 2, 3 and 4 are determined uniformly between one and five, where each of them is placed one, two or three times (with equal probabilities) on the PCB. The rest of the components were in group-1 that are composed of 40 component types and the placement number of each one is probabilistically equal. For the larger problems (*N*=200, 300 or 400), the number of component types in each group is kept constant but their placement numbers are increased proportionally with *N*. The board that these components are placed is assumed to have dimensions of 250 mm by 300 mm and it is assured that no two components are placed on the same coordinate.

An investigation of the design of placement locations of the heavier components reveals the fact that the components in the same group are usually placed closely. Their placement locations are mostly very close to each other with one or two of them being far away from the group. That is, the distribution of heavy components throughout the board is heterogeneous. Thus, we have two board models at hand, the board type in which all components are randomly distributed on the PCB and the model in which heavier components are placed closely. In this study, we will call the former one as homogeneous model and the latter one as structured model. To obtain structured problem instances the data generator is re-implemented to give PCB instances in which heavier groups of components are placed closely. This is achieved by allocating a region for group-2 components on the southwest side of the PCB, a region for group-3 components on the west side of the PCB and a region for group-4 components on the northwest side of the PCB. On the other hand, group-1 components are randomly distributed on the PCB.

For each PCB model, four data sets of 100 instances having 100, 200, 300 and 400 components respectively, are produced by using data generator and they are kept in files as inputs for the algorithms. In this section, any value appearing in a comparison table is the average of 100 PCBs in specified PCB model having specified number of components.

V.2. COMPARISON CRITERIA

The heuristics developed for PCB assembly can be compared by using the following criteria.

V.2.1. Total assembly time

The basic and most important comparison criterion in PCB placement machines is the total assembly time of a given PCB. The assembly time values are recorded in units of seconds.

V.2.2. Lower Bound

Lower Bound (LB) is an important term and can be used as a comparison factor. The reader should remember that SC_i is defined as the number of rotational steps of the turret in a speed category i and formulations used for deriving SC_i values are different for ATMA and iATMA. If we assume that the x-y movements of the board carrier can be completed within the allowed turret time, tt_i, then, the sum of the 'number of placements in a speed category i times the corresponding tt_i values' plus $N*t^0$ can be defined as the LB for the PCB assembly time. Clearly, the LB values of the algorithms are comparison criteria. We can define LB mathematically as follows.

$$LB = N \times t^0 + \sum_{i=1}^{K} SC_i \times tt_i \qquad \text{(V.1)}$$

When the above data generation model is analyzed, one can easily see that in a typical PCB with 100 components (*N=100*), number of types of components and number of components to be placed for each group (N_i and n_i values) can be expected as follows: n_1=40, n_2=3, n_3=3, n_4=3, N_1=82, N_2=6, N_3=6 and N_4=6. Applying the related formulas for calculating number of rotational steps of the turret in a speed category *i,* (SC_i) values for ATMA and iATMA can be obtained easily (Table V.1).

Table V.1—LB values for both algorithms on a PCB with 100 components

i	n_i	N_i	SC_i *(ATMA)*	SC_i *(iATMA)*
1	40	82	56	62
2	3	6	9	6
3	3	6	9	6
4	3	6	26	26
Lower Bound			31.64	31.16

By applying equation (V.I), the lower bound for this board is computed as 56*0.25 + 9*0.28 + 9*0.38 + 26*0.45 = 31.64 seconds for ATMA and 31.16 for iATMA which is calculated in a similar fashion. This means that if all *x-y* movements of the board carrier are completed within turret time, 1.5% improvement on the average can be obtained.

The same analysis can be applied when *N*=400 and one can obtain the results in Table V.2.

Table V.2—LB values for both algorithms on a PCB with 400 components

i	n_i	N_i	SC_i *(ATMA)*	SC_i *(iATMA)*
1	40	340	314	320
2	3	20	23	20
3	3	20	23	20
4	3	20	40	40
Lower Bound			111.68	111.20

Thus, if all placements can be done within turret time, then iATMA is expected to outperform ATMA in terms of total assembly time.

It is interesting to note that the improvement amount that iATMA generates remains constant when *N* increases, and so relative improvement decreases to 0.4%. The deviation from lower bound values are also given in the tables in the following sections which are also stated in units of seconds.

V.2.3. Run Time

Another criterion is the running time complexity of the proposed algorithms but for a continuous manufacturing system, time needed for designing the placement sequence (running time of algorithm) appears to be negligible when compared with total production time. However, for the sake of a detailed analysis, the run time of the algorithms will also be indicated. The run time values are recorded in units of seconds.

The tests are performed on a machine with Intel Core2 CPU at 2.0 GHz with 2 GB RAM using Windows Vista OS.

V.3. ATMA vs iATMA

The comparison results of ATMA and iATMA for the homogeneous are given in Table V.3.

Table V.3—Assembly Time and LB values for both algorithms (homogeneous boards)

N	ATMA		iATMA	
	Assembly Time	Lower Bound	Assembly Time	Lower Bound
100	40.55	31.53	40.20	31.07
200	66.55	58.19	67.06	57.72
300	92.21	84.80	92.68	84.33
400	118.13	111.17	118.76	110.70

As indicated before, the lower bound value for iATMA is less than ATMA's lower bound value for all values of *N*. But the results are not totally in favor of iATMA. They show that even though iATMA is better than ATMA in terms of lower bound values, it did not produce lower assembly times than ATMA except for *N*=100. So, as a result, it is proved that smaller lower bounds can be obtained by placing groups of components in the reverse order (heaviest to lightest) but iATMA failed at reaching these lower bounds and even gave worse results. The possible cause of this situation is that in iATMA we face with the case $CF1(i, j) > tt_i$ for consecutive placement operations more than we face with it in ATMA. That is, the travel time from placement location to another is greater than turret time for more number of cases than predicted. If this travel time were smaller than tt_i, then we could achieve lower assembly times. The extended analysis given below clarifies this discussion.

For each group, number of components placed in each speed category can be summarized in Table V.4 and Table V.5 for ATMA and iATMA. The considered PCB board is the typical PCB board with 100 components described above.

Table V.4—Number of components placed in each SC for each group in ATMA

ATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 82	56	9	9	8
Group-2	Out of 6	-	-	-	6
Group-3	Out of 6	-	-	-	6
Group-4	Out of 6	-	-	-	6

Table V.5—Number of components placed in each SC for each group in iATMA

iATMA		SC_1	SC_2	SC_3	SC_4
Group-1	Out of 82	62	-	-	20
Group-2	Out of 6	-	6	-	-
Group-3	Out of 6	-	-	6	-
Group-4	Out of 6	-	-	-	6

In homogeneous boards, when generating the locations of components on a board the only consideration that we take into account is not to place two components on the same location. So the components in a group are randomly distributed over the board. When group-1 components are randomly distributed on the board and their route is built, it is seen that the distance between two placements have reasonable length. That is, for most cases the travel time for this distance is less than or comparable to the turret time. But when other group components are randomly distributed on the board and the TSP route for them is found, it is seen that the distance between two placements do have considerably larger length. That is, for most cases the travel time of the board carrier for this distance can be greater than the turret time. So they cannot be placed in the turret time and an excess time is unavoidable. In the first table above (Table V.4), we see that ATMA places the group-2, 3 and 4 components when the turret time is at maximum (during SC_4). But iATMA places group-2 components during SC_2 and group-3 components during SC_3, which is when turret time is smaller. So, ATMA compensates the possible excess time more effectively than iATMA, hence obtains a really big advantage. On the other hand, iATMA places more number of group-1 components in SC_4 than ATMA. But this will imply a small benefit for iATMA because, for most cases the travel time for this distance will not be much greater than the turret time.

To summarize, to minimize the excess time occurring while placing the group-2, 3 and 4 components, placing them in minimum speed (SC_4) brings an advantage to ATMA and so it performs better.

On the other hand, on structured boards, ATMA is expected to give worse results than iATMA. Structured boards contain heavier components that are located closer to each other. Thus, placement time for heavier components could be less than turret time values. Hence, the disadvantage for iATMA disappears in structured board instances. In Table V.6, we give the results of the comparison of the PCB assembly time obtained by ATMA and iATMA. Lower bound values for homogeneous and structured boards are different because different random PCBs are produced for each PCB model.

Table V.6—Assembly Time and LB values for both algorithms (structured boards)

N	ATMA		iATMA	
	Assembly Time	Lower Bound	Assembly Time	Lower Bound
100	35.83	31.68	34.58	31.19
200	61.00	58.23	60.01	57.76
300	86.45	84.53	85.43	84.06
400	113.15	111.50	112.03	111.02

iATMA outperforms ATMA by 3.50% for N=100, by 1.63% for N=200, by 1.18% for N=300 and by 0.99% for N=400. Improvement percentages decrease as N increases. This result could be expected because as N increases, the average x-y distances between component pairs decreases and it will be easier to arrange the placement sequence so that the board carrier movements can be completed within the free (turret) time for both iATMA and ATMA.

In the following sections, the performance of the proposed heuristics are analyzed. Some of these heuristics require an initial solution of the problem and some of them can be applied after another. The consecutively applied heuristics will be indicated by a "+" between them. For example, ATMA+AFPP+RRTLEM indicates that initial solution is obtained by ATMA, then AFPP is used to improve the solution which is followed by RRTLEM.

V.4. ADJUST FIRST POINT PROCEDURE (AFPP AND AFPTCP)

The performances of AFPP and AFPTCP are compared with ATMA. Recall that the difference between these two heuristics is that in AFPP, placement cost of only group-1 components is used whereas, AFPTCP decides on adjustment based on placement cost of all components. The results are promising as given in (Table V.7).

Table V.7—Performance of AFPP and AFPTCP on boards with 100 components

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
ATMA	40.55	-	9.02	0.53	35.83	-	4.15	0.51
ATMA+AFPP	39.92	1.55%	8.39	0.54	36.12	-0.79%	4.44	0.55
ATMA+AFPTCP	38.62	4.77%	7.09	0.54	34.38	4.06%	2.70	0.55
iATMA	40.20	0.85%	9.13	0.50	34.58	3.50%	3.39	0.52
iATMA+AFPP	40.44	0.27%	9.37	0.51	35.89	-0.15%	4.70	0.51
iATMA+AFPTCP	39.21	3.32%	8.14	0.51	34.34	4.17%	3.15	0.50

In Table V.7, results for both homogeneous and structured boards are given for PCBs with 100 components. Improvement values are obtained by comparing the assembly time values by ATMA. For example applying AFPTCP after ATMA on homogeneous boards improves the result by 4.77 percent, while applying it after iATMA improves the solution created by ATMA by 3.32 percent. On the other hand, deviation from lower bound values are obtained by subtracting lower bound values of ATMA or accordingly iATMA (given in Table V.3 and Table V.6) from the assembly time values. For example, applying AFPTCP after ATMA on homogeneous boards gives an assembly time value which is 7.09 seconds away from ATMA's lower bound, whereas applying AFPTCP after iATMA gives an assembly time value which is 8.14 seconds away from iATMA's lower bound. We should note, in the rest of this study, the improvement and deviation from lower bound values should be interpreted in the way explained above.

It is interesting to note that AFPP demonstrates a poor performance and even deteriorates the placement sequence given by ATMA. This is because of the poor design of the algorithm in which a starting point of assembly process (first point of group-1) is reassigned to a new point if the cost of placing group-1 components is minimized. However, minimizing assembly cost of group-1 components does not necessarily reduce the total assembly cost. This deficiency is properly handled in AFPTCP and decision for making a point as the first component for the assembly is based on two adjustments; first rearranging all tours and secondly total cost of assembly.

As we pointed out before, success of AFPTCP is due to two reasons. In the first one, AFPTCP looks for improvement opportunities that minimize total assembly cost in cases where better (closer) connection points exist. Consider the following example given in Figure V.1 and Figure V.2.

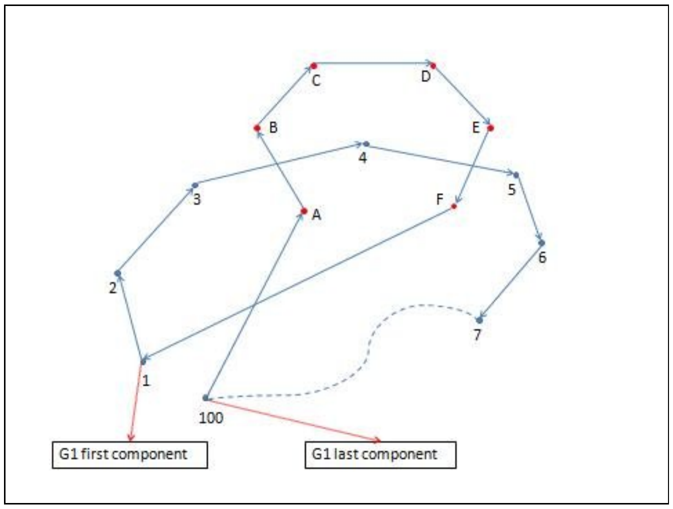


Figure V.1—An example depicting solution of ATMA

In this example, for simplicity, assume that we have two groups of components and number of group-1 components is 100 (labelled numerically) while number of group-2 components is six (labelled with characters from A to F). The solution given in Figure V.1 is given by ATMA and the assembly starts from the southwest component, component 1. After the tour of group-1 components is finished, assembly continues from the group-2 component closest to the last component of group-1, component 100. When assembly of group-2 components is complete, assembly of a new board starts over from component 1. It is easily seen that distance between connection points increases assembly cost.

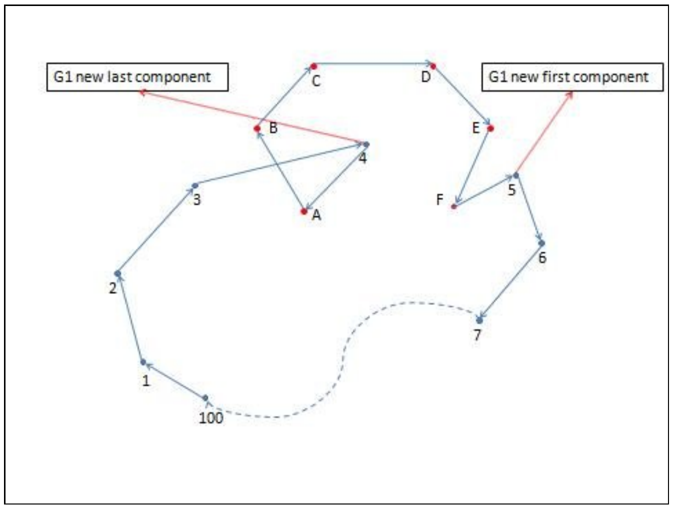


Figure V.2—An example showing improvement of AFPP

On the contrary, we observe small distances between connection points in Figure V.2. This cost reduction opportunity is caught by AFPP while trying new starting points for the assembly.

Secondly, starting the assembly from a different point may cause more number of placements to be done within turret time. As an example, consider the following TSP tour with two turret time values. Assume that tt_1 is 0.20 s, tt_2 is 0.40 s and tt_1 is used as the parameter in *CF2* for the first five placements and tt_2 is used for the rest.

$$A \xrightarrow[0.5]{} B \xrightarrow[0.40]{} C \xrightarrow[0.30]{} D \xrightarrow[0.20]{} E \xrightarrow[0.40]{} F \xrightarrow[0.20]{} G \xrightarrow[0.30]{} H \xrightarrow[0.10]{} I \xrightarrow[0.20]{} J \xrightarrow[0.30]{} A$$

Suppose also that *CF1(x, y)* values are given between each pair of components. If the starting point for the tour is selected as *A*, then total travel cost is calculated as *CF2*(*A*, *B*, 0.20) + *CF2*(*B*, *C*, 0.20) + *CF2*(*C*, *D*, 0.20) + *CF2*(*D*, *E*, 0.20) + *CF2*(*E*, *F*, 0.20) + *CF2*(*F*, *G*, 0.40) + *CF2*(*G*, *H*, 0.40) + *CF2*(*H*, *I*, 0.40) + *CF2*(*I*, *J*, 0.40) + *CF2*(*J*, *A*, 0.40) = 0.5 + 0.4 + 0.3 + 0.2 + 0.4 + 0.4 + 0.4 + 0.4 + 0.4 + 0.4 = 3.8 s. On the other hand, if the starting point is selected as *F*, then total assembly cost is calculated as *CF2*(*F*, *G*, 0.20) + *CF2*(*G*, *H*, 0.20) + *CF2*(*H*, *I*, 0.20) + *CF2*(*I*, *J*, 0.20) + *CF2*(*J*, *A*, 0.20) + *CF2*(*A*, *B*, 0.40) + *CF2*(*B*, *C*, 0.40) + *CF2*(*C*, *D*, 0.40) + *CF2*(*D*, *E*, 0.40) + *CF2*(*E*, *F*, 0.40) = 0.2 + 0.3 + 0.2 + 0.2 + 0.3 + 0.5 + 0.4 + 0.4 + 0.4 + 0.4

= 3.3 seconds. Hence, new improvements for assembling group-1 components can be obtained by a different preference of starting point.

Another observation from the results is about the lower bound values. It is easily seen that better performing heuristics show smaller deviations from lower bound and we see that the smallest deviation from lower bound is obtained by AFPTCP when applied after ATMA.

As a result, AFPTCP is an efficient heuristic that outperforms ATMA algorithm by more than 4.50 percent on homogeneous boards. On the other hand, AFPTCP decreases assembly time to 34.34 seconds (smallest assembly time on structured boards) when applied after iATMA.

V.5. POSTPONE DEVIANT

Performance of PD and EPD heuristics are compared and the following results in Table V.8 are obtained.

Table V.8—Performance of PD and EPD on boards with 100 components

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
ATMA+PD	38.73	4.50%	7.20	3.88	35.01	2.30%	3.33	4.03
ATMA+EPD	38.51	5.05%	6.98	10.85	34.94	2.49%	3.26	9.58
ATMA+AFPTCP+PD	37.26	8.12%	5.73	3.80	34.12	4.80%	2.44	4.05
ATMA+AFPTCP+EPD	37.11	8.48%	5.58	10.56	34.07	4.92%	2.39	9.33
iATMA+PD	38.76	4.42%	7.69	4.02	34.38	4.04%	3.19	3.99
iATMA+EPD	38.63	4.75%	7.56	10.35	34.36	4.11%	3.17	8.10
iATMA+AFPTCP+PD	37.80	6.78%	6.73	3.79	34.17	4.65%	2.98	4.29
iATMA+AFPTCP+EPD	37.67	7.10%	6.60	10.43	34.15	4.69%	2.96	7.56

It is easily observed that, when compared with PD, EPD brings further improvements in terms of assembly time using about three times of run time, as expected due to the nature of their design. Another observation is that improvement percentage stays at a lower value on structured boards. In structured boards the components in groups other than group-1 are more closely located and so their placement operation is mostly completed within turret time. Hence, little improvement opportunity is caught.

When EPD is applied after AFPTCP, the gained improvement reaches up to 8.48 percent on homogeneous boards. As a result, combination of two or more heuristics brings better assembly time values and this reality will be more apparent in the following sections.

V.6. INDIVIDUAL INSERTION AND GROUP INSERTION HEURISTICS

In Table V.9, we show the performance comparison of the II and GI heuristics applied after ATMA on both homogeneous and structured PCBs with 100 components (*N*=100). The improvement values indicate the amount of percentage that ATMA is improved. From the table it can be seen that on structured boards, using II provides a significant improvement when compared with GI. When the same comparison is done by considering homogeneous boards, improvements of two insertion techniques stay almost at the same amount.

Table V.9—Performance of PD and EPD on boards with 100 components when applied after ATMA

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
ATMA+II	39.93	1.54%	8.40	0.55	34.74	3.06%	3.06	0.53
ATMA+GI	39.88	1.66%	8.35	0.54	35.22	1.72%	3.54	0.56

In Table V.10, we show the performance improvements of the II and GI heuristics applied after iATMA. The values indicate the performance gain with respect to iATMA. When compared with the previously analyzed heuristics, performance of II and GI is worse than AFPP or AFPTCP. However, their runtime is comparably smaller than PD or EPD.

Table V.10—Performance of PD and EPD on boards with 100 components when applied after iATMA

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
iATMA+II	40.16	0.96%	9.09	0.55	34.31	4.24%	3.12	0.56
iATMA+GI	39.86	1.70%	8.79	0.55	34.34	4.16%	3.15	0.56

V.7. RECORD-TO-RECORD WITH LOCAL EXCHANGE MOVES

As described previously, the RRTLEM heuristic uses two parameters that affect its performance heavily. They are the number of iterations (*noi*) that the general framework includes and percentage value (*rate*) that is used in the local search moves. Using a high *noi* value definitely enlarges the search space, but causes an increase in the run time proportionally. On the other hand, a low *rate* value may not give the opportunity to escape from local minima, and a high *rate* value will cause the algorithm traverse on the inefficient peak points. Thus a reasonable choice of these parameters is necessary in order to get maximum performance, and this can be done only after an extended analysis.

V.7.1. Number of Iterations parameter (*noi*)

Firstly, *noi* parameter is analyzed on structured boards. We set the *rate* parameter as 1%. The results are given in Table V.11.

Table V.11—Analysis of *noi* parameter (*rate*=1%, structured boards)

noi	Assembly Time	Run Time (sec)	Performance Improvement
1	34.85	0.59	2.75%
3	34.7	0.63	3.15%
5	34.61	0.68	3.42%
10	34.5	0.85	3.72%
20	34.43	1.16	3.93%
30	34.38	1.40	4.07%
50	34.34	1.97	4.16%
100	34.31	3.47	4.26%
200	34.27	6.42	4.36%
400	34.25	11.85	4.43%
800	34.23	23.05	4.47%
1600	34.2245	45.91	4.49%
3200	34.2199	88.95	4.50%
6400	34.2196	183.34	4.51%
12500	34.2196	348.30	4.51%

We see that, increasing *noi* increases the run time of the algorithm as expected. Also the assembly time decreases by an increase in *noi*. It is important to note that applying 6400 iterations or 12500 iterations results in the same value in the assembly time. Hence the performance improvement obtained by increasing the *noi* parameter converges at 6400. However, the run time increases proportionally with *noi*, reaching up to more than three minutes. In the table one can easily see that the run time for *noi*=400 takes almost 12 seconds and the performance improvement according to ATMA is 4.43%, where the run time for *noi*=6400 takes almost three minutes and the performance improvement is 4.51%. So, according to us, it is rational to a have shorter run time for the sake of giving up 0.08% performance improvement. Thus we chose to use 400 as the *noi* value in our future research.

V.7.2. Rate parameter

After deciding on the *noi* parameter, we concentrate on the *rate* parameter of our algorithm. After a little investigation, we saw that the *rate* parameter should not be smaller than 0.1%. Figure V.3 gives details of our search applied on structured boards. This time *noi* is taken as 400 as explained above.

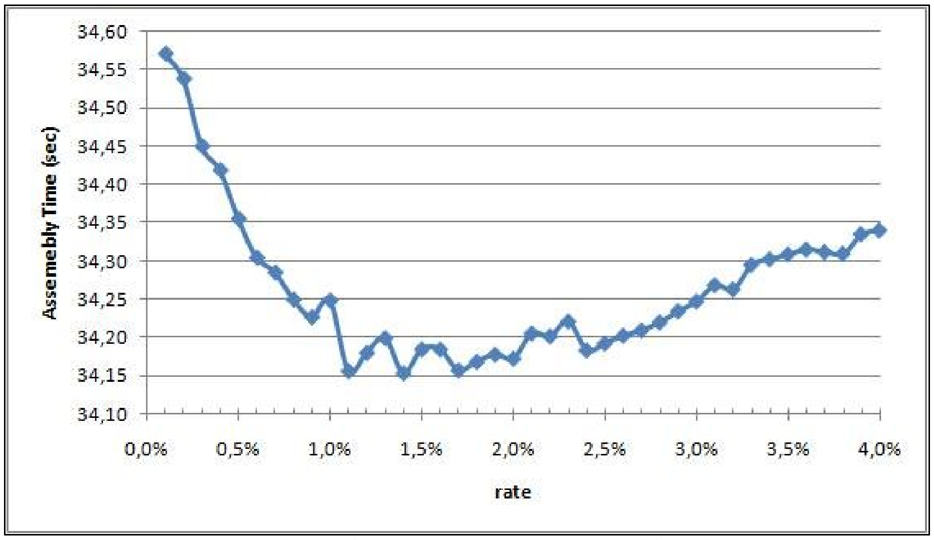


Figure V.3—Analysis of *rate* parameter on structured boards. (*noi*=400)

We can analyze the results in Figure V.3 in three parts as follows. The figure shows a continuous decrease from 0.1% to 0.9%. Between 1% and 2% we observe a steady state with small fluctuations and for *rate* values larger than 2% almost a continuous increase is the case. For values smaller than 1%, we can say that the algorithm cannot enlarge the search space, so it has smaller chance to escape from local minima. For values larger than 2%, the search space is enlarged more than necessary and so the algorithm can get far away from the good solutions. For values between 1% and 2% the algorithm focuses on effective places where better solutions are maintained. We give the assembly time values for the *rate* between 1% and 2% in Table V.12. In this table, we observe that the *rate* parameter shows best performance when it is equal to 1.4%. By this analysis, we show that in order to increase the performance of the algorithm, rather than using a higher *noi* value, one should prefer to choose a more effective *rate* value. That is, when *rate* is equal to 1%, we increased the *noi* parameter up to 6400 which is almost 15 times more costly in terms of run time, but we gained only a 0.08% performance improvement. However, by setting *rate* parameter equal to 1.4%, we obtained 0.26% performance improvement, where run time stays constant.

Table V.12—Analysis of *rate* parameter on homogeneous boards. (*noi*=400)

rate	Assembly Time	Performance Improvement
0.01	34.25	4.43%
0.011	34.16	4.68%
0.012	34.18	4.62%
0.013	34.2	4.56%
0.014	34.15	4.69%
0.015	34.18	4.60%
0.016	34.18	4.60%
0.017	34.16	4.68%
0.018	34.17	4.65%
0.019	34.18	4.62%
0.02	34.17	4.64%

The same analysis is made for homogenous boards and we reached almost the same results (Figure V.4). Smallest assembly time is obtained when *rate* parameter is equal to 1.2% and 1.4% (Table V.13).

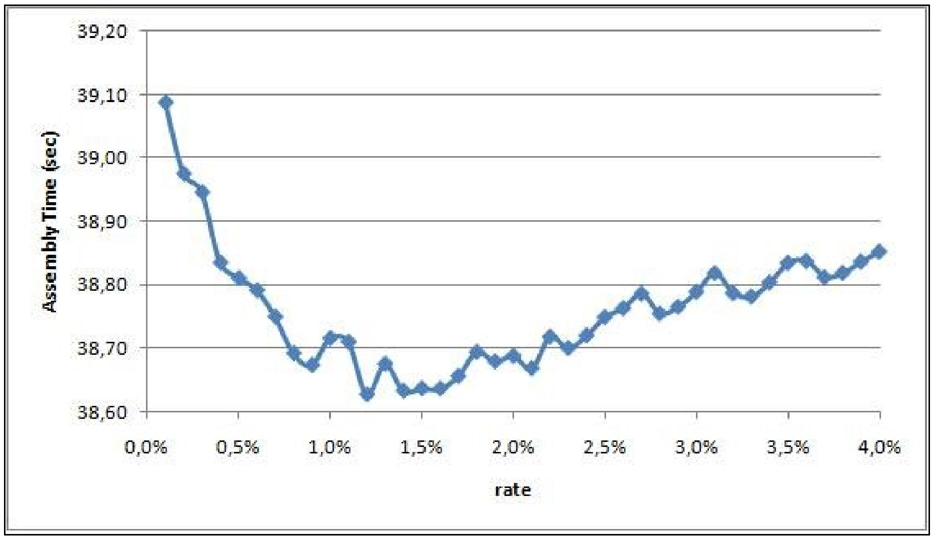


Figure V.4—Analysis of *rate* parameter on homogeneous boards. (*noi*=400)

Thus we can state that, in our future research, we will use the *rate* parameter as 1.4% and *noi* parameter as 400.

Table V.13—Analysis of *rate* parameter on homogeneous boards. (*noi*=400)

rate	Assembly Time	Performance Improvement
0.01	38.72	4.53%
0.011	38.71	4.54%
0.012	38.63	4.75%
0.013	38.68	4.63%
0.014	38.63	4.73%
0.015	38.64	4.72%
0.016	38.64	4.72%
0.017	38.66	4.67%
0.018	38.69	4.58%
0.019	38.68	4.62%
0.02	38.69	4.60%

Since RRTLEM is an improvement heuristic based on local search moves, it can be applied after any other algorithm. The performance of RRTLEM and application of it after ATMA and combination with other heuristics are given in Table V.14.

Table V.14—Performance of RRTLEM and other heuristics on boards with 100 components when applied after ATMA

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
ATMA+RRTLEM	38.63	4.73%	7.10	11.74	34.15	4.69%	2.47	11.37
ATMA+RRTLEM+PD	37.24	8.17%	5.71	15.10	33.83	5.59%	2.15	15.06
ATMA+RRTLEM+EPD	37.11	8.48%	5.58	20.92	33.77	5.75%	2.09	20.55
ATMA+AFPTCP+RRTLEM	37.97	6.37%	6.44	11.55	33.91	5.38%	2.23	11.36
ATMA+AFPTCP+RRTLEM+PD	36.80	9.26%	5.27	14.97	33.69	5.99%	2.01	14.79
ATMA+AFPTCP+RRTLEM+EPD	36.70	9.51%	5.17	20.95	33.65	6.08%	1.97	19.70
ATMA+II+RRTLEM	38.54	4.97%	7.01	11.86	34.06	4.96%	2.38	11.52
ATMA+GI+RRTLEM	38.52	5.00%	6.99	11.73	34.03	5.04%	2.35	11.51

Results regarding RRTLEM indicate that it is a powerful heuristic that brings at least 4.73 percent and up to 9.51 percent of improvement when applied after

ATMA. However, it has a very large runtime. Especially when it is combined with EPD, the time for obtaining a placement sequence and feeder configuration reaches up to 20 seconds. It is also worth mention that the deviation from lower bound decreases to 1.97 s in the best case.

Similar discussions are also valid when RRTLEM is applied after iATMA (Table V.15). The best improvement occurs when it is applied after iATMA with AFPTCP and EPD which is about 8.58 percent.

Table V.15—Performance of RRTLEM and other heuristics on boards with 100 components when applied after iATMA

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
iATMA+RRTLEM	38.95	3.96%	7.88	10.83	33.75	2.40%	2.56	10.44
iATMA+RRTLEM+PD	37.56	7.37%	6.49	14.05	33.58	6.28%	2.39	13.91
iATMA+RRTLEM+EPD	37.44	7.67%	6.37	20.05	33.57	6.31%	2.38	18.25
iATMA+AFPTCP+RRTLEM	38.48	5.11%	7.41	10.55	33.77	5.76%	2.58	10.43
iATMA+AFPTCP+RRTLEM+PD	37.18	8.31%	6.11	13.91	33.60	6.23%	2.41	13.77
iATMA+AFPTCP+RRTLEM+EPD	37.07	8.58%	6.00	19.47	33.58	6.28%	2.39	17.34
iATMA+II+RRTLEM	38.89	4.10%	7.82	10.46	33.71	5.92%	2.52	10.38
iATMA+GI+RRTLEM	38.85	4.19%	7.78	10.69	33.71	5.92%	2.52	10.43

V.8. PAIRWISE EXCHANGE PROCEDURE (PEP)

The results of applying PEP after ATMA and iATMA along with their comparison discussion are as follows.

Table V.16—Improvement ratios of PEP when applied after ATMA and iATMA (structured board)

Heuristic	Assembly Time	Improvement			NOX	NOCI	Run Time
		Average	Max	Min			
ATMA+PEP	35.30	1.50%	5.06%	0.30%	5326	100	17.54
iATMA+PEP	34.36	0.63%	2.45%	0.00%	7254	93	15.69

In Table V.16, we show the performance comparison of the algorithms on structured model PCBs with 100 components (N=100). It is important to keep in mind that each *assembly time*, *average improvement* and *NOX* value in the table is the average of 100 different PCB instances. *Max* column denotes the maximum improvement (percentage) value; *Min* column denotes the minimum improvement (percentage) value observed among 100 instances. The number of exchange trials set for PEP is 10^6, and *NOX* denotes the number of successful exchanges. *NOCI* denotes the number cases (instances) in which ATMA (or iATMA) is improved by using PEP, out of 100 instances. For example, ATMA+PEP improved ATMA in all cases while iATMA+PEP outperformed iATMA in 93 cases and in the remaining seven they gave the same result.

From a general point of view, among all algorithms in Table V.16, we observe that ATMA is improved by iATMA from 35.83 to 34.58 which is improved by PEP to 34.36 according to total assembly time, on the average. This brings about a 4.11% improvement, in total.

For integrity, the behavior of PEP on homogenous PCB model is studied and presented in Table V.17.

Table V.17—Improvement ratios of PEP when applied after ATMA and iATMA (homogeneous board)

Heuristic	Ass. Time	Improvement			NOX	NOCI
		Average	Max	Min		
ATMA+PEP	39.35	2.97%	8.39%	0.00%	5056	99
iATMA+PEP	39.72	1.21%	3.96%	0.00%	6894	95

Thus, we show that applying PEP after ATMA or iATMA brings significant improvements to total assembly times of PCBs. We proved that in almost 95% of randomly generated PCBs, an improvement is possible.

Since PEP is an improvement heuristic based on local search move, it can be applied after any other algorithm. The performance of PEP and application of it after other heuristics are given in Table V.18.

PEP requires run time of at least 15 seconds to produce a placement sequence. When compared with RRTLEM, PEP shows inferior performance because it gives worse assembly time in a greater amount of time. However, it is a simple and straightforward algorithm when compared with RRTLEM.

Table V.18—Performance of PEP and other heuristics on boards with 100 components with respect to ATMA

	Homogeneous Boards				Structured Boards			
Heuristic	Assembly Time	Imp.	Dev. from LB	Run Time	Assembly Time	Imp.	Dev. from LB	Run Time
ATMA+PEP	39.35	2.97%	7.82	18.02	35.30	1.50%	3.62	17.54
ATMA+AFPTCP+PEP	38.47	5.13%	6.94	17.52	34.30	4.28%	2.62	17.72
ATMA+II+PEP	39.70	2.10%	8.17	17.60	34.64	3.34%	2.96	17.54
ATMA+GI+PEP	39.46	2.69%	7.93	17.74	34.78	2.95%	3.10	17.60
iATMA+PEP	39.72	2.06%	8.65	15.76	34.36	4.11%	3.17	15.69
iATMA+AFPTCP+PEP	39.05	3.70%	7.98	16.14	34.24	4.46%	3.05	16.83
iATMA+II+PEP	39.91	1.59%	8.84	15.74	34.24	4.46%	3.05	15.87
iATMA+GI+PEP	39.52	2.54%	8.45	15.94	34.21	4.52%	3.02	15.84

V.9. SUMMARY OF RESULTS

In this chapter, we investigated the performance of the proposed heuristics in the thesis. The performance of the heuristics is compared with the previously proposed algorithms.

When the heuristics are compared in terms of run time, the best are iATMA and AFPTCP. iATMA improves ATMA by 3.50 percent on structured PCBs whereas the improvement percent stays at 0.85 on homogeneous boards. On the other hand, AFPTCP minimizes the assembly time by 4.77 percent on structured boards. II and GI have a runtime comparable with iATMA and AFPTCP, however their performance is worse.

When the heuristics are compared in terms of total assembly time, PD, RRTLEM, AFPTCP and EPD show the best performance. Their performance improvement on homogeneous boards is 4.50 percent, 4.73 percent, 4.77 percent and 5.05 percent, respectively. However, RRTLEM improves ATMA by 4.69 percent, AFPTCP by 4.06 percent, EPD by 2.49 percent and PD by 2.30 percent on structured boards. Among these heuristics, AFPTCP requires the smallest amount of time while RRTLEM requires the most.

Combining two heuristics mostly improves a given solution further. For example, the best combination pair is applying RRTLEM and EPD after ATMA. It improves ATMA by 8.48 percent on homogeneous boards and by 5.75 percent on structured boards. Another well performing pair is AFPTCP and EPD which also shows 8.48 percent improvement on homogeneous boards.

If more than two heuristics are applied after ATMA, then the performance improvement reaches up to 9.51 percent. We see this level of improvement when AFPTCP, RRTLEM and EPD are successively applied after ATMA. However, we should note that the runtime also increases with additional heuristics applied.

The best assembly time on homogeneous boards is obtained by ATMA+AFPTCP+RRTLEM+EPD which is 36.70. On the other hand, the best assembly time on structured boards is obtained by iATMA+RRTLEM+EPD combination of heuristics by 33.57.

As a result, in this study several heuristics with different characteristics are proposed. Almost all of them outperform previously proposed approaches.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

VI.1. RESEARCH WORK SUMMARY AND CONTRIBUTIONS

Within this technology era, printed circuit boards (PCBs) are used in almost every technological device. Due to the wide usage area of PCBs, one of the crucial ways to increase a PCB manufacturer company's competitiveness is to minimize the production time and thus reduce the costs. To attain this goal, the component placement process must be optimized because it is generally the bottleneck of a PCB assembly line. Since the performance of the placement machines studied in this thesis depends heavily on both the placement sequencing and feeder configuration problems, they were studied and solved. The significant achievements of this thesis are summarized in the following:

1. A comprehensive survey on chip shooter placement machines was carried out. The literature regarding the chip shooter machines are categorized according to the solution approaches adopted. It is figured out that several heuristics are developed for solving these problems altogether but all of them are based either on GA or on applying iterative improvements by applying several heuristics on separate problems. Machine specifications for several machine models are discussed in detail. Varying turret rotation time is an important property of chip shooters but very few of the studies propose solution procedures considering this property.
2. For both placement machines, including the chip mounter machine and chip shooter machine, the placement sequencing problem by taking into account the varying turret rotation time have been formulated as nonlinear integer programming model. We pointed out that the problem arising due to varying rotational turret time values of a chip shooter is actually a new generalization of the TSP. To the best of our knowledge this TSP generalization is not

defined in the literature and we named it as the Sequence Dependent TSP (SDTSP).

3. For chip mounter placement machine, we formulated an original linear integer programming model for the feeder configuration problem. Besides, we formulated an original nonlinear integer programming model for the assembly time minimization problem (integrated problem of placement sequencing and feeder configuration). A secondary problem is formulated for chip mounter machines. The problem arises when a previously proposed heuristic, Assembly Time Minimization Algorithm (ATMA), is applied to the assembly time minimization problem. It is shown that the secondary problem is actually a TDTSP with special cost functions.
4. For chip shooter placement machine, we formulated an original nonlinear integer programming model for the feeder configuration problem. Besides, in the study, an original nonlinear integer programming model is formulated for the assembly time minimization problem (integrated problem of placement sequencing and feeder configuration) for these machines.
5. Various heuristics are proposed and applied to optimize the assembly process of chip mounter machine.

   Inverse ATMA (iATMA) is a heuristic that improves the assembly time of chip mounter machines. It simply reverses the assembly process of component weight groups and thus obtains new improvement opportunities for structured PCBs.

   For minimizing the assembly time of PCBs, another approach we developed is based on trying different first (starting) points for the assembly process. Of course, this is applicable to the cases in which the preference of first point of the tour does not matter and in placement machines this condition is satisfied. Adjust First Point Procedure (AFPP) and Adjust First Point according to Total Cost Procedure (AFPTCP) are developed in this understanding.

   The general idea in the studies that considers components of different weights is to group components according to their turret speed values, build a TSP tour for each group and place these groups of components from lightest to heaviest or vice versa. Postpone Deviant (PD) or Extended PD (EPD) mixes components of different weight categories if it minimizes total assembly time.

Two local search heuristics are applied to the assembly time minimization problem. Record-to-Record with Local Exchange Moves (RRTLEM) is a local search method that combines Record-to-Record (RRT) and local exchange moves. Pairwise Exchange Procedure (PEP) starts with a given solution and applies a number of random pair-wise (1-1) exchanges.

The performance of these heuristics are tested on randomly generated PCB data. The results show that up to nine percent improvement is obtained when compared with previous approaches used for chip mounter machines.

VI.2. FUTURE WORK

The analyzed machines in this study are commonly used machine types. They are thoroughly studied within the thesis. However, there are other types of machines that are newly gaining acceptance in the industry. One of these machines is the collect and place machine. We believe that a study regarding its mathematical formulation would be a valuable contribution to the literature.

We observe that several heuristics are developed for solving these problems altogether but all of them are based either on GA or iterative solution methodology. Application of other meta-heuristics such as Simulated Annealing (SA), Ant Colony Optimization (ACO) and etc. is not detected in the literature. However, these meta-heuristics are applied for solving combinatorial optimization problems and for some of them they are shown to demonstrate better performances than GA. Hence, application of these meta-heuristics seems to fill up a gap in this research area. On the other hand, component retrieval problem is not addressed as much as the feeder configuration and placement sequencing problems.

One of the assumptions that we made for the thesis is that number of component types is equal to the number of feeder slots on the machine. The studied problems can be reformulated by considering feeder duplications. This will also yield the optimization of component retrieval problem.

The ultimate purpose of research is to apply the research results to industry. The placement sequencing problem, feeder configuration problem and the integrated assembly time minimization problem are from the PCB manufacturing industry, so it is reasonable to return the research solutions to the industry in such a way that the productivity can be improved.

REFERENCES

[1] Ahmadi, J.; Ahmadi, R.; Matsuo, H.; Tirupati, D.: “Component fixture positioning for printed circuit board assembly with concurrent operations”, *Operations Research*, 43, **(1995)** 444-457.

[2] Albiach, J.; Sanchis, J. M.; Soler, D.: “An asymmetric TSP with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation”, *European Journal of Operational Research*, 189, **(2008)** 789–802.

[3] Alkaya, A.F.; Duman, E.: “Assembly Time Minimization of a Particular Placement Machine”, *Proceedings of the 12th WSEAS International Conference on Applied Mathematics*, Cairo, Egypt. December 29-31, **(2007)**, 383-388.

[4] Alkaya, A.F.; Duman, E.; Eyler, M.A: “Assembly time minimization for an electronic component placement machine”, *WSEAS Transactions on Computers*, 7, **(2008)** 326-340.

[5] Alkaya, A.F.; Duman, E.: “Improving the efficiency of an electronic component placement machine”, *Abstract Book of the 2nd International Conference on Control and Optimization with Industrial Applications*, Baku, Azerbaijan, June 2-4, **(2008)** 35.

[6] Alkaya, A.F.; Duman, E.: “Heuristics for a generalization of TSP in the context of PCB assembly”, *Proceedings of the International Conference on Prospects for Research in Transport and Logistics on a Regional Global Perspective*, Istanbul, Türkiye, February 12-14, **(2009)**, 167-172.

[7] Alkaya, A.F.; Duman, E.: “A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines”, *Proceedings of PICMET’09*, Portland, OR, USA, August 2-6, **(2009)**, 3296-3306.

[8] Ammons, J.C.; Carlyle, M.; Depuy, G. W.; Ellis, K.P.; McGinnis, L.F.; Tovey, C.A.; Xu, H.: “Computer-aided process planning in printed circuit card assembly”, *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, 16, **(1993)** 370-376.

[9] Ayob, M.; Kendall, G.: “A survey of surface mount device placement machine optimisation: Machine classification”, *European Journal of Operational Research*, 186, **(2008)** 893–914.

[10] Balas, E.; Ceria, S.; Cornuejols, G.: “Mixed 0-1 programming by lift-and-project in a branch-and-cut framework”, *Management Science*, 42, **(1996a)** 1129-1246.

[11] Balas, E.; Ceria, S.; Cornuejols, G.; Natraj, N.: “Gomory Cuts revisited”, *Operations Research Letters*, 19, **(1996b)** 1-9.

[12] Bard, J.F.; Clayton, R.W.; Feo, T.A.: “Machine setup and component placement in printed circuit board assembly”, *International Journal of Flexible Manufacturing Systems*, 6, **(1994)** 5-3.

[13] Beasley, D.; Bull, D. R.; Martin R. R.: “An overview of genetic algorithms: Part I, Fundamentals”, *University Computing*, 15, **(1993)** 58-69.

[14] Bigras, L.-P.; Gamache, M.; Savard, G.: “The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times”, *Discrete Optimization*, 5, **(2008)** 685-699.

[15] Blum, C.; Roli, A.: “Metaheuristics in combinatorial optimization: Overview and conceptual comparison”, *ACM Computing Surveys*, 35, **(2003)** 268-308.

[16] Boyd, E.A.; “Fenchel Cutting Planes for Integer Programs”, *Operations Research*, 42, **(1994)** 53-64.

[17] Bozer, Y.A.; Shorn, E.C.; Sharp, G.P.: “Geometric approaches to solve chebyshev traveling salesman problem”, *IIE Transactions*, 22, **(1990)** 238-254.

[18] Chen, W.S.; Chyu, C.C.: “Reduction of printed circuit board group assembly time through the consideration of efficiency loss of placement time", *Assembly Automation*, 22, **(2002)** 360-370.

[19] Choi, E.; Tcha, D.: “A column generation approach to the heterogeneous fleet vehicle routing problem”, *Computers & OR*, 34, **(2007)** 2080-2095.

[20] Chyu, C.C.; Chang, W.S.: “A genetic-based algorithm for the operational sequence of a high speed chip placement machine”, *International Journal of Advanced Manufacturing* Technology, 36, **(2008)** 918-926.

[21] Clausen, J.: “Branch and Bound Algorithms - Principles and Examples”, *Universitetsparken 1, DK-2100*, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, **(1999)**.

[22] Crama, Y.; Kolen, A. W. J.; Oerlemans, A. G.; Spieksma, F. C. R.: “Throughput Rate Optimization in the Automated Assembly of Printed Circuit Boards”, *Annals of Operations Research*, 26, **(1990)** 455-480.

[23] Crama, Y.; Flippo, O.E.; Klundert, J.V.D.; Spieksma, F.C.R.: “The Component Retrieval Problem in Printed Circuit Board Assembly”, *International Journal of Flexible Manufacturing Systems*, 8, **(1996)** 287-312.

[24] Crama, Y.; Flippo, O.E.; Klundert, J.V.D.; Spieksma, F.C.R.: “The assembly of printed circuit boards: a case with multiple machines and multiple board types”, *European Journal of Operational Research*, 98, **(1997)** 457-472.

[25] Crama, Y.; Klundert, J.V.D.; Spieksma, F.C.R.: “Production Planning Problems in Printed Circuit Board Assembly”, *Discrete Applied Mathematics*, 123, **(2002)** 339-361.

[26] Croes, G.: “A Method for Solving Traveling-Salesman Problems”, *Operations Research*, 6, **(1958)** 791-812.

[27] Csaszar, P.; Tirpak, T. M.; Nelson, P. C.: “Optimization of a high-speed placement machine using tabu search algorithms”, *Annals of Operations Research*, 96, **(2000)** 125-147.

[28] Dantzig, G.B.: “Maximization of a linear function subject to linear inequalities” in T.C. Koopmans (ed.), *Activity Analysis of Production and Allocation*, John Wiley and Sons, New York, **(1951)** 317-329.

[29] Dantzig, G.B.; Ramser J.H.: “The truck dispatching problem”, *Management Science*, 6, **(1959)** 80-91.

[30] De Souza, R.; Lijun, W.: “CPS: A productivity tool for component placement in multi-head concurrent operation PCBA machines”, *Journal of Electronics Manufacturing*, 4, **(1994)** 71-79.

[31] De Souza, R.; Lijun, W.: “Intelligent Optimization of Component Insertion in Multi-Head Concurrent Operation PCBA Machines”, *Journal of Intelligent Manufacturing*, 6, **(1995)** 235-243.

[32] Dikos, A.; Nelson, P.C.; Tirpak, T.M.; Wang, W.: "Optimization of high-mix printed circuit card assembly using genetic algorithms", *Annals of Operations Research*, 75, **(1997)** 303–324.

[33] Drysdale, C.: "Report: IC Packaging Topped $30B in 2007", *Circuits Assembly, The Journal for Mount and Electronics Assembly*, **(2008)**, http://circuitsassembly.com/cms/content/view/6587/95/ (15.05.2008)

[34] Dueck, G.: "New optimization heuristics: the great deluge algorithm and the record-to-record travel", *Journal of Computational Physics*, 104, **(1993)** 86-92.

[35] Duman, E.: "Optimization Issues in Automated Assembly of Printed Circuit Boards", *PhD Thesis*, Boğaziçi University, Istanbul, Türkiye, **(1998)**.

[36] Duman E.; Or I.: "Precedence Constrained TSP Arising in Printed Circuit Board Assembly", *International Journal of Production Research*, 42, **(2004)** 67-78.

[37] Duman, E.; Yıldırım, M.B.: "Workload Balancing in Printed Circuit Board Assembly Shops", *Doğuş University Journal*, 6, **(2005)** 67-78.

[38] Duman E.: "Modelling the operations of a component placement machine with rotational turret and stationary component magazine", *Journal of the Operational Research Society*, 58, **(2007a)** 317-325.

[39] Duman E.: "Constrained multiple TSP in the context of printed circuit board assembly", *Proceedings of the International Conference on Industrial Engineering and Systems Management*, Beijing, China, **(2007b)** 9-13.

[40] Duman, E.; Or, I.: "The quadratic assignment problem in the context of the printed circuit board assembly process", *Computers & OR*, 34, **(2007)** 163-179.

[41] Duman, E.: "An Application of the Multiple TSP in Printed Circuit Board Assembly", *Journal of Operations and Logistics*, 2 (3), **(2009)**, III.1-III.9

[42] Duman, E.; Alkaya, A.F.: "A New Generalization of the Travelling Salesman Problem" to be presented in *III. Congress of the Turkish World Mathematicians*, Almaty, Kazakhstan, June 30–July 04, **(2009)**.

[43] Ellis, K.P.; Vittes, F.J.; Kobza, J.E.: "Optimizing the Performance of a Surface Mount Placement Machine", IEEE Transactions on Electronics Packaging Manufacturing, 24, **(2001)** 160-170.

[44] Fox, K.; Gavish, B.; Graves, S.C.: "An n-Constraint Formulation of the (Time Dependent) Traveling Salesman Problem", *Operations Research*, 28, **(1980)** 1019–1021.

[45] Gavish, B.; Seidmann, A.: "Printed Circuit Boards Assembly Automation-Formulations ans Algorithms", *Proceedings of Ixth ICPR*, Cincinnati, OH, USA, August 17-20, **(1987)**.

[46] Gendrau, M.; Laporte, G.; Musaraganyi, C.; Taillard, E.D.: "A tabu search heuristic for the heterogeneous fleet vehicle routing problem", *Computers & OR*, 26, **(1999)** 1153-1173.

[47] Golden, B.L.: "Large-Scale Vehide Routing and Related Combinatorial Problems" , *PhD Thesis*, MIT Operations Research Center, Cambridge, Mass.,USA, **(1976).**

[48] Golden, B.L.; Assad, A.A.; Levy L.; Gheysens F.G.: "The Fleet Size and Mix Vehicle Routing Problem", *Computers & OR*, 11, **(1984)** 49-66.

[49] Golden, B.L.; Wasil, E.A.; Kelly, J.P.; Chao, I.M.: "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results", In: Crainic T.; Laporte G.: editors. *Fleet management and logistics*. Boston, MA: Kluwer, **(1998)** 33–56.

[50] Gomory, R.E.: "Outline of an algorithm for integer solutions to linear programs", *Bulletin of the American Mathematical Society*, 64, **(1958)** 275-278.

[51] Gomory, R.E.; "An algorrithm for integer solutions to linear programs", *Recent Advances in Mathematical Programming*, R.L. Graves; P.Wolfe Eds. McGraw-Hill, New York, **(1963)** 269-302.

[52] Grunow, M.; Gunther, H.O.; Fohrenbach A.: "Simulation-based performance analysis and optimization of electronics assembly equipment", *International Journal of Production Research*, 38, **(2000)** 4247-4259.

[53] Gutin, G.; Punnen, A.: *The Traveling Salesman Problem and its Variants*, Kluwer Academic Publishers, **(2002)**.

[54] Haghani, A.; Jung, S.: "A dynamic vehicle routing problem with time-dependent travel times", *Computers & Operations Research*, 32, **(2005)** 2959–2986.

[55] Hanan, M.; Kurtzberg, J.M.: "A review of the placement and quadratic assignment problems", *SIAM Review*, 14, **(1972)** 324-342.

[56] Hitchcock, F. L.: "The distribution of a product from several sources to numerous localities", *Journal of Mathematics and Physics*, 20, **(1941)** 224-230.

[57] Ho, W.; Ji, P.: "Component scheduling for chip shooter machines: a hybrid genetic algorithm approach", *Computers & OR*, 30, **(2003)** 2175-2189.

[58] Ho, W.; Ji, P.: "A hybrid genetic algorithm for sequencing PCB component placement", *International Journal of Knowledge-based and Intelligent Engineering Systems*, 9, **(2005)** 129-136.

[59] Ho, W.; Ji, P.: "A genetic algorithm approach to optimising component placement and retrieval sequence for chip shooter machines", *International Journal of Advanced Manufacturing Technology*, 28, **(2006)** 556-560.

[60] Ho, W.; Ji, P.: "Optimal Production Planning for PCB Assembly", *Springer Series in Advanced Manufacturing*, Springer-Verlag London Limited **(2007)**.

[61] Ho, W.; Ji, P.; Wu, Y.: "A heuristic approach for component scheduling on a high-speed PCB assembly machine", *Production Planning & Control*, 18, **(2007)** 655-665.

[62] Holland, H: "Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems", in Machine Learning: An Artificial Intelligence Approach, Volume II, R. S. Michalski et al., Eds. Los Altos, CA: Morgan Kaufmann, **(1986)** 593-623.

[63] Horak, T.; Francis, R.: "Utilization of machine characteristics in PC board assembly", *Research paper*, MSCIS Dept., Rutgers University, USA, **(1995)**.

[64] Ichoua, S.; Gendreau, M.; Potvin J.Y.: "Vehicle dispatching with time-dependent travel times", *European Journal of Operational Research*, 144, **(2003)** 379–396.

[65] Jeevan, K.; Parthiban, A.; Seetharamu, K. N.; Azid, I. A.; Quadir, G. A.: "Optimization of PCB Component Placement using Genetic Algorithms", *Journal of Electronics Manufacturing*, 11, **(2002)** 69-79.

[66] Jensen, P.A.; Bard, J.F.: *Operations Research: Models and Methods*, Wiley, New York, **(2003)**.

[67] Jünger, M.; Thienel, S.; Reinelt, G.: "Provably good solutions for the traveling salesman problem", *Mathematical Methods of Operations Research*, 40, **(1994)** 183-217.

[68] Jünger, M.; Reinelt, G.; Thienel, S.: "Practical problem solving with cutting plane algorithms in combinatorial optimization", *Combinatorial Optimization: DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, *AMS*, **(1995)** 111-152.

[69] Karmarkar, N.: "A New Polynomial Time Algorithm for Linear Programming", *Combinatorica*, 4, **(1984)** 373-395.

[70] Khoo, L.P.; Loh, K.M.: "A genetic algorithms enhanced planning system for surface mount PCB assembly", *International Journal of Advanced Manufacturing Technology*, 16, **(2000)** 289-296.

[71] Klomp, C.; Klundert, J. J. V. D.; Spieksma, F. C. R.; Voogt, S.: "The feeder rack assignment problem in pcb assembly: a case study", *International Journal of Production Economics*, 64, **(2000)** 399-407.

[72] Kumar, R.; Luo, Z.: "Optimizing the Operation Sequence of a Chip Placement Machine Using TSP Model", *IEEE Transactions on Electronics Packaging Manufacturing*, 26, **(2003)** 14-21.

[73] Laporte, G.: "The vehicle routing problem: An overview of exact and approximate algorithms", *European Journal of Operational Research*, 59, **(1992)** 345-358.

[74] Leipala, T.; Nevalainen O.: "Optimization of the Movement of a Component Placement Machine", *European Journal of Operational Research*, 38, **(1989)** 167-177.

[75] Letchford, A.N.; Lodi, A.: "Strengthening Chavatal-Gomory Cuts and Gomory fractional cuts", *Operations Reseach Letters*, 30, **(2002)** 74-82.

[76] Leu, M.C.; Wong, H., Ji. Z.: "Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm", *Journal of Electronic Packaging*, 115, **(1993)** 424-432.

[77] Li, F.; Golden, B.; Wasil, E.: "Very large-scale vehicle routing: new test problems, algorithms, and results", *Computers & OR*, 32, **(2005)** 1165-1179.

[78] Li, F.; Golden, B.; Wasil, E.: “A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem”, *Computers & OR*, 34, **(2007)** 2734-2742.

[79] Lima, C.M.M.R.; Goldbarg, M.C.; Goldbarg, E.F.G.: “A memetic algorithm for the heterogeneous fleet vehicle routing problem”, *Electronic Notes in Discrete Mathematics*, 18, **(2004)** 171-176.

[80] Lustig, I.; Marsten, R.E.; Shanno, D.F.: “Interior point methods for linear programming: computational state of the art”, *Report SOR 92-17*, Program in Statistics and Operations Research, Department of Civil Engineering and Operations Research, Princeton University, **(1992)**.

[81] Magyar, G.; Johnsson, M.; Nevalainen, O.: “On solving single machine optimization problems in electronics assembly”, *Journal of Electronics Manufacturing*, 9, **(1999)** 249-267.

[82] Malandraki, C.; Daskin, M.S.: “Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms”, *Transportation Science*, 26, **(1992)** 185-200.

[83] Marchand, H.; Martin, A.; Weismantel, R.; Wolsey, L.: “Cutting planes in integer and mixed integer programming”, *Discrete Applied Mathematics*, 123, **(2002)** 397-446.

[84] 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)** 18-29.

[85] Mitchell, J.E.: “Branch-and-Cut Algorithms for Combinatorial Optimization Problems”, *Handbook of Applied Optimization*, Oxford University Press, January, **(2002)** 65-77.

[86] Moyer, L.K.; Gupta, S.M.: “SMT feeder slot assignment for predetermined component placement paths”, *Journal of Electronics Manufacturing*, 6, **(1996a)** 173-192.

[87] Moyer, L.K.; Gupta, S.M.: “Simultaneous Component Sequencing and Feeder Assignment for High Speed Chip Shooter Machines”, *Journal of Electronics Manufacturing*, 6, **(1996b)** 271-305.

[88] Moyer, L.K.; Gupta, S.M.: "An Efficient Assembly Sequencing Heuristic for Printed Circuit Boards Configurations", *Journal of Electronics Manufacturing*, 7, **(1997)** 143-160.

[89] Nelson, K.M.; Wille, L.T.: "Comparative study of heuristics for optimal printed circuit board assembly", *Conference Record for Southcon*, **(1995)** 322-327.

[90] Ng, M. K.: "Heuristics Approach to Printed Circuit Board Insertion Problem", *Journal of the Operational Research Society*, 49, **(1998)** 1051-1059.

[91] Ng, M. K.: "Clustering methods for printed circuit board insertion problems", *Journal of the Operational Research Society*, 51, **(2000)** 1205-1211.

[92] Ochi, L.S.; Vianna D.S.; Drummond, L.M.A.; Victor, A.O.: "A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet", *Future Generation Computer Systems*, 14, **(1998)** 285-292.

[93] Ohno, K.; Jin, Z.; Elmaghraby, S.E.: "An optimal assembly mode of multi-type printed circuit boards", *Computers & IE*, 36, **(1999)** 451-471.

[94] Ong, N.S.; Tan, W.C.: "Sequence placement planning for high-speed PCB assembly machine", *Integrated Manufacturing Systems*, 13, **(2002)** 35-46.

[95] Or, I.: "Traveling Salesman Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking", *PhD Thesis*, Northwestern University, **(1976)**.

[96] Osman, I.H.; Kelly, J.P: *Meta-heuristics: Theory & Applications*, Kluwer Academic Publishers, Boston, **(1996)**.

[97] Osman, I.; Salhi, S.: "Local search strategies for the vehicle fleet mix problem", Rayward-Smith, V.J.; Osman, I.H.; Reeves, C.R.; Smith, G.D. (Eds.), *Modern Heuristic Search Methods*, Wiley, New York, **(1996)** 131–153.

[98] Padberg, M.; Rinaldi, G.: "A branch-and-cut algorithm for the resolution large-scale symmetric traveling salesman problem", *SIAM Review*, 33, **(1991)** 60-100.

[99] Picard, J.C.; Queyranne, M.: "The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling", *Operations Research*, 26, **(1978)** 86-110.

[100] Renaud, J.; Boctor, F.F.: “A sweep-based algorithm for the fleet size and mix vehicle routing problem”, *European Journal of Operational Research*, 140, **(2002)** 618-628.

[101] Schneider, J.: “The time-dependent traveling salesman problem”, *Physica A*, 314, **(2002)** 151–155.

[102] Shih, W.; Srihari, K.; Adriance, J.: “Expert system based placement sequence identification for surface mount PCB assembly”, *International Journal of Advanced Manufacturing Technology*, 11, **(1996)** 413-424.

[103] Smed, J.; Johnsson, M.; Nevalainen O.: “A hierarchical classification scheme for electronics assembly problems”, *Proceedings of TOOLMET Symposium—Tool Environments and Developments Methods for Intelligent Systems*, Oulu, Finland, **(2000)** 116–119.

[104] Sohn, J.; Park, S.: “Efficient Operation of a Surface Mounting Machine with a Multihead Turret”, *International Journal of Production Research*, 34, **(1996)** 1131-1143.

[105] Stewart, W.R.: “A computationally efficient heuristic for the traveling salesman problem”, *Proceedings of the 13th Annual Meeting of Southeastern TIMS*, Myrtle Beach, SC, USA, **(1977)** 75–83.

[106] Taha, H.A.: *Operations Research: An Introduction*, Prentice Hall, New Jersey, **(2003).**

[107] Taillard E.D.: “A heuristic Column Generation Method for the Heterogeneous Fleet VRP”, *RAIRO*, 33, **(1999)** 1-14.

[108] Tarantilis, C.D.; Kiranoudis, C.T.; Vassiliadis, V.S.: “A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem”, *Journal of the Operational Research Society*, 54, **(2003)** 65-71.

[109] Tarantilis, C.D.; Kiranoudis, C.T.; Vassiliadis, V.S.: “A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem”, *European Journal of Operational Research*, 152, **(2004)** 148-158.

[110] Tarantilis, C.D.; Kiranoudis, C.T.: “A flexible adaptive memory-based algorithm for real-life transportation operations: Two case studies from dairy and construction sector”, *European Journal of Operational Research*, 179, **(2007)** 806-822.

[111] Tirpak, T.M.: “Design to manufacturing information management for electronics assembly”, *International Journal of Flexible Manufacturing Systems*, 12, **(2000)** 189-205.

[112] Toth, P.; Vigo, D.: “Models, relaxations and exact approaches for the capacitated vehicle routing problem”, *Discrete Applied Mathematics*, 123, **(2002)** 487-512.

[113] Vittes, F.J.: “Optimizing the performance of a chip shooter machine”, *MS Thesis*, Ind. Sys. Eng. Dept., Virginia Tech, Blacksburg, USA **(1999)**.

[114] Wang, W.; Nelson, P.C.; Tirpak,T.M.: “Optimization of High-Speed Multistation SMT Placement Machines Using Evolutionary Algorithms”, *IEEE Transactions on Electronics Packaging Manufacturing*, 22, **(1999)** 137-146.

[115] Yeo, S.H.; Low, C.W.; Yong, K.H.: “A rule-based frame system for concurrent assembly machines”, *International Journal of Advanced Manufacturing Technology*, 12, **(1996)** 370-376.

[116] Walas, R.A.; Askin, R.G.: “An algorithm for NC turret punch press tool location and hit sequencing”, *IIE Transactions*, 16, **(1984)** 280–287.

[117] Waters, C.D.J.: “A Solution Procedure for the Vehicle-Scheduling Problem Based on Iterative Route Improvement”, *The Journal of the Operational Research Society*, 38, **(1987)** 833-839.

[118] Wiel, R.J.V.; Sahinidis, N.V.: “Heuristic Bounds and Test Problem Generation for the Time Dependent Traveling Salesman Problem”, *Transportation Science*, 29, **(1995)** 167-183.

[119] Wiel, R.J.V.; Sahinidis, N.V.: “An exact solution approach for the time-dependent traveling-salesman problem”, *Naval Research Logistics*, 43, **(1996)** 797-820.

[120] Williams, H.P.: *Model Building in Mathematical Programming*, John Wiley & Sons, New York, **(1999)**.

[121] Winston, W.L.; Venkataramanan, M.: *Introduction to Mathematical Programming: Operations Research*, Brooks/Cole-Thomson Learning, California, **(2003)**

RESUME

ALI FUAT ALKAYA
Eğitim Mah. Patika Sok.
Yıldırım Apt. B Blok No: 10 D: 5
Kadıköy ISTANBUL
Phone (H)+90-216-345-3867, (Cell)+90-532-502-0689 (W)+90-216-348-0292 ext 253
e-mail: falkaya@marmara.edu.tr, falkaya@gmail.com

EDUCATION

Ph.D Engineering Management	Marmara University, Faculty of Engineering, Istanbul, Turkey, 2009. *Thesis topic*: "Optimizing the Operations of Electronic Component Placement Machines", *Advisor*: Prof. Dr. M. Akif Eyler, *Co-Advisor*: Assoc. Prof. Dr. Ekrem Duman.
M.S., Computer Science	Marmara University, Faculty of Engineering, Istanbul, Turkey, 2002. *Thesis Topic:* Task Scheduling Techniques for Arbitrary Network Topologies, *Advisor*: Assoc. Prof. Dr. Haluk R. Topçuoğlu
B.S., Mathematics	Koç University, Istanbul, Turkey, 1998.

WORK EXPERIENCE

1998-…	*Research Assistant*, in Department of Computer Science, Marmara University, Istanbul, Turkey.
2000-…	*Lecturer:* CSE141 Introduction to Computer Programming (JAVA), CSE142 Intermediate Programming, Math259 Numerical Analysis, MATH255 Discrete Mathematics, C Programming, Computer Programming with Pascal in Department of Computer Science, Marmara University, Istanbul, Turkey. *Teaching Assistant*: CSE225 Data Structures, CSE461 Compiler Design, CSE364 Principles of Programming Languages, MATH257 Linear Algebra, CSE474 Computer Networks, CSE100 Introduction to Computer Engineering, CSE200 C Programming in Department of Computer Science, Marmara University, Istanbul, Turkey.
2004-2007	*Adjunct lecturer* for several courses (Calculus, Discrete Mathematics, Management Information Systems, Data Structures and Algorithm Analysis, Operations Research) in Preston University, Istanbul, Turkey
2002-2004	*Adjunct lecturer* for "Internet and Web Programming" and "Information Technologies" courses in Beykent University, Istanbul, Turkey.
2001-2002	*CCNA Instructor,* Marmara University Local Academy.
1999-2000	*System Administrator,* NT 4.0, Windows 2000 Advanced Server
Summer 1997	*Lab Assistant,* CIT Department, Koç University, Istanbul, Turkey.

PROFESSIONAL SKILLS

English: fluent (TOEFL : 600)
German: fair
GRE: old scores (1998): 780, 360, 710 (Q, V, A scores, respectively)
GRE:790, 360, 2.5 (Q, V, A scores, respectively)
Experienced in MS Office (Word, Excel, PowerPoint)
MS Access, MS Visual Basic, Fortran
MS Visual C++, C++ Builder, Turbo C, ASP
Delphi, Turbo Pascal, Matlab
C Programming in Unix, Linux environment.
JAVA Programming within several IDE environments (Eclipse, NetBeans, etc.)

GRADUATE COURSES and PROJECTS

Parallel Processing, Computer Design, Compiler Theory, Computational Complexity, Data Structures, Digital Design, Algorithm Analysis and Design, Fuzzy Logic, Object Oriented Design, Object Oriented Programming with Java, Learning Automata, Management Information Systems, Strategic Management, .

2008-...	*Graduate Student Scholarship,* Active role in proposal and progress of TUBITAK 1001 project titled as "Optimization of Electronic Assembly Operations and Sequence Dependent Traveling Salesman Problem" (Project ID:108M198)
2008	*Technical Committee Member,* Design of Student Affairs Automation System
2004	*Technical Committee Member,* improving the network infrastructure of Marmara University (a million $ budget project)

RESEARCH INTERESTS

Combinatorial Optimization, Operations Research, PCB Placement Machines, Networking, Task Scheduling and Parallel Processing, Routing Algorithms, Statistics, Artificial Intelligence, Fuzzy Logic, Computational Theory, Compiler Theory, Computational Applications of Math to Business and Information Systems, Software Development & Information Systems

FOREIGN LANGUAGES

English (fluent), German (fair)

AWARDS AND HONORS

Appreciation Certificate from Institute of Graduate Studies in Pure and Applied Sciences due to SCI publication from MS thesis, 2006
Dean's Honor Roll, Koç University, Fall 1997
Entered Koç University with full scholarship, 1994.
In the first 1000 among 1.5 million test-takers in the University Entrance Exam 1994
Entered Fatih High School, one of the best schools in Turkey, with full scholarship, 1987

PUBLICATIONS (Journal)

Duman, E; Yıldırım, M.B.; Alkaya, A.F.: “Scheduling continuous aluminium casting lines”, *International Journal of Production Research*, 46, (20), **(2008)**, 5701-5718.

Alkaya, A.F.; Duman, E.; Eyler, M.A.:“Assembly time minimization for an electronic component placement machine”, *WSEAS Transactions on Computers*, 7, (4), **(2008)**, 326-340.

Alkaya, A. F.; Topçuoğlu, H. R.: “A task scheduling algorithm for arbitrarily-connected processors with awareness of link contention”, *Cluster Computing*, 9, **(2006),** 417-431.

PUBLICATIONS (Conference Proceedings)

Duman, E.; Alkaya, A.F.: “defining a new variant of the Traveling Salesman Problem”, accepted for *AAS 2009 Conference on Applied Automatic Systems*, Ohrid, Macedonia, September 26-29, **(2009)**.

Alkaya, A.F.; Duman, E.: “A Literature Survey of the Operation Optimization in Chip Shooter Placement Machines”, *Proceedings of PICMET’09*, Portland, OR, USA, August 2-6, **(2009)**, 3296-3306.

Alkaya, A.F.; Duman, E.: “Heuristics for a generalization of TSP in the context of PCB assembly”, *Proceedings of the International Conference on Prospects for Research in Transport and Logistics on a Regional Global Perspective*, Istanbul, Türkiye, February 12-14, **(2009)**, 167-172.

Alkaya, A.F.; Duman, E.: “Assembly Time Minimization of a Particular Placement Machine”, *Proceedings of the 12th WSEAS International Conference on Applied Mathematics*, Cairo, Egypt. December 29-31, **(2007)**, 383-388.

PUBLICATIONS (Conference Presentation)

Alkaya, A.F.; Duman, E.: “Improving the efficiency of an electronic component placement machine”, presented in *2nd International Conference on Control and Optimization with Industrial Applications*, Baku, Azerbaijan, June 2-4, **(2008)**.

Duman, E.; Alkaya, A.F. “A New Generalization of the Traveling Salesman Problem” presented in *III. Congress of the Turkish World Mathematicians*, Almaty, Kazakhstan, June 30–July 04, **(2009)**

PUBLICATIONS (Other)

MS Frontpage 2000 Lecture Notes (Prepared for Web Design course).