The Hong Kong Polytechnic University

Department of Industrial and Systems Engineering

Optimization of Production Planning in Printed Circuit Board Assembly

WU Yongzhong

A thesis submitted in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

October 2008

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

______________________________(Signed)

__________**Wu Yongzhong**________(Name of student)

ABSTRACT

Many complicated planning problems arise in Printed Circuit Board (PCB) assembly. This research focuses on two high-level planning problems, i.e., the Component Allocation Problem (CAP) and the Multi-Line Scheduling Problem (MLSP), both of which are important for improving production efficiency in PCB assembly.

For a PCB job (batch) to be processed by an assembly line, the component allocation problem is investigated, which is to allocate the component placements required by the PCB to the placement machines in the line, so that the line cycle time is minimized. The problem is intertwined with the lower-level machine optimization problems (feeder arrangement and placement sequencing), which determine the process (placement) time of each machine. Considering the great computational complexity, a decomposed solution strategy is proposed. This strategy relies on a regression-based placement time estimator, which can estimate the placement time of each machine accurately without solving the machine optimization problems. Based on this estimator, a specific genetic algorithm is developed. Experimental tests show that the proposed genetic algorithm can solve the problem both effectively and efficiently. Compared with the existing software provided by the machine vendor, the line cycle time is reduced.

For a set of PCB jobs to be produced by multiple assembly lines, the multi-line scheduling problem is investigated, which is to assign the PCB jobs to the lines and sequence the jobs in each line, so that the sum of weighted tardiness and weighted makespan is minimized. A mixed integer linear programming model for the problem is established. Line-dependent cycle times, different due dates of the jobs, sequence-dependent setup times, and precedence constraints are considered so that the model is realistic and applicable. Experimental tests show that exact solutions can not be obtained for realistic-sized problem instances. A specific genetic algorithm is developed for solving the problem. Due to the complexity of the problem, a new replacement strategy is proposed to improve the performance of the algorithm. Experimental tests show that the genetic algorithm can solve the problem both effectively and efficiently. A study of a real case is conducted and illustrates the applicability and usefulness of the method.

ACKNOWLEDGMENTS

This research would have not been so successful without the help of many people who took great support. I would like to take this opportunity to express my heartfelt gratitude towards my chief supervisor Dr P. Ji., whose tireless inspiration and help guided me to overcome the difficulties in my PhD study and my life. His encouragement and valuable suggestions have significantly helped the development of the project.

Besides, I have to express my sincere gratitude to my family (my wife, Peng Cui, in particular), and friends for their tremendous support and encouragement.

Definitely, the financial support from The Hong Kong Polytechnic University enables this project possible. I would like to acknowledge all parties from the heart including those mentioned above together with those who helped me indeed but missed to thank before.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

With rapid development of advanced technologies, electronic products and devices can be found in our major life activities of working, living and learning, and they significantly affect the quality of life. As the backbone of most electronic products, Printed Circuit Boards (PCBs) have become much smaller in size and more densely populated with components. As a result, the Surface Mount Technology (SMT) has replaced the Pin-through-Hole (PTH) technology since the SMT enables to mount a large number of electronic components on a small board. With the technological advances of the SMT, the PCB assembly process has evolved from a labor-intensive activity to a highly automated one. However, the great expense of the assembly machines and the customers' demands on both speed and quality motivate PCB manufacturers to optimize the assembly operations and achieve high production efficiency.

1.1 OVERVIEW OF PCB ASSEMBLY

The PCB assembly process in the SMT environment consists of four main operations in sequence, i.e., solder paste application, component placement, solder paste reflow, and inspection. Figure 1.1 illustrates a typical PCB assembly line in the PCB manufacturing industry.

First, on the screen printer, solder paste is "printed" to the places where the components will be put on the board. Then, the required components are placed onto the board by the placement machine(s). After that, the PCB with the components placed on it is conveyed through the reflow oven, which causes the solder paste to reflow and form the solder joints. Finally, inspection is performed at the inspection station.

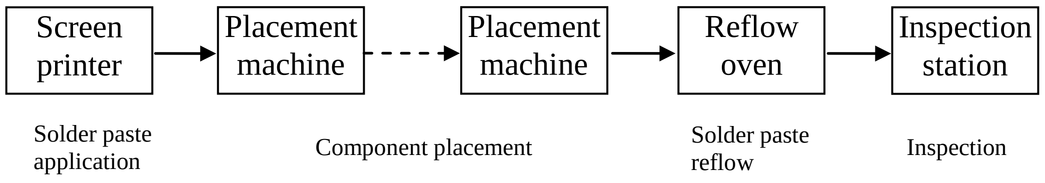


Figure 1.1 A typical PCB assembly line

Among these operations, the component placement process is most time-consuming. For this reason, there may be more than one placement machine in the line. In spite of this, the placement process is always the bottleneck process in most cases. For this reason, most planning problems in PCB assembly focus on the placement process.

Modern PCB placement machines (as illustrated in Figure 1.2) are much sophisticated and very expensive, with each ranging from US $300,000 to $1,000,000. In a typical PCB assembly shop, there may be several assembly lines with dozens of placement machines, for a total value of several million US dollars.

The placement process conducted by the placement machines is highly automatic. To complete a placement job, the component feeders, each supplying components of a specific type, are set up in the feeder carrier of the machine. After the PCB is loaded onto the PCB holder, the placing device on the machine begins to pick the components from the feeders and place them onto the board. Section 3.4.1 will describe the placement process of a placement machine in more details. Depending on the operating mode of the pick-and-place process, the placement machines can be divided into various types, e.g., *sequential pick-and-place*, *turret-type*, *collect-and-place*, *dual-delivery*, and *multi-station*.

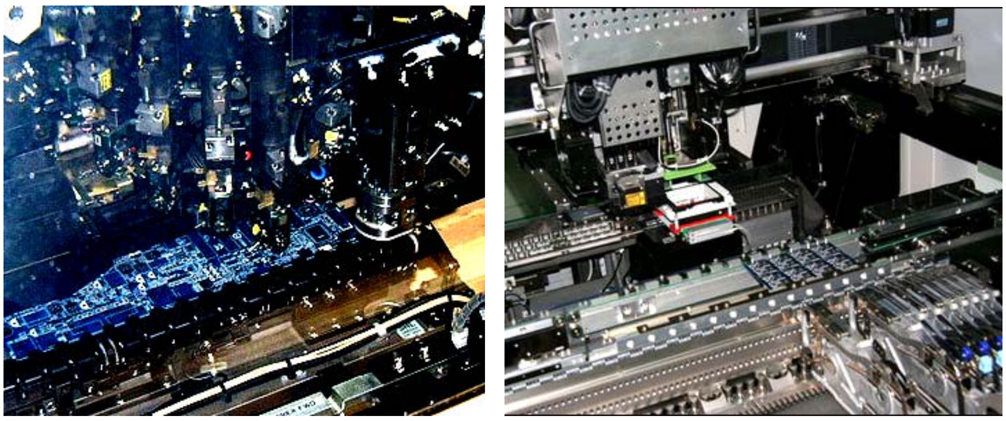


Figure 1.2 PCB placement machines

Over the planning horizon, many batches of PCB are required to be produced. They are assigned to different assembly lines for production. An assembly line usually produces the batches in a flowshop manner, i.e., a new batch of boards can only be produced on the line after the completion of the previous batch.

Due to the wide range of both components and products involved, the complexity of the placement machines, and the volatility of customer demand, it is extremely difficult to solve the planning problems in multi-line SMT facilities so as to improve production efficiency and customer satisfaction at the same time.

1.2 PLANNING PROBLEMS IN PCB ASSEMBLY

1.2.1 Planning Hierarchy

The planning problems in PCB assembly may vary owing to the great variety of production environments concerning shop layout, production mix, setup policy, etc. Nevertheless, there are three planning problems that are common for many multi-line PCB manufacturers, i.e., the Multi-Line Scheduling Problem (MLSP), the Component Allocation Problem (CAP) or the Line Balancing Problem, and the Machine Optimization Problem (MOP). Figure 1.3 illustrates the relationship between these problems in the planning hierarchy. A similar discussion was presented by Ammons *et al*. [Amm97].

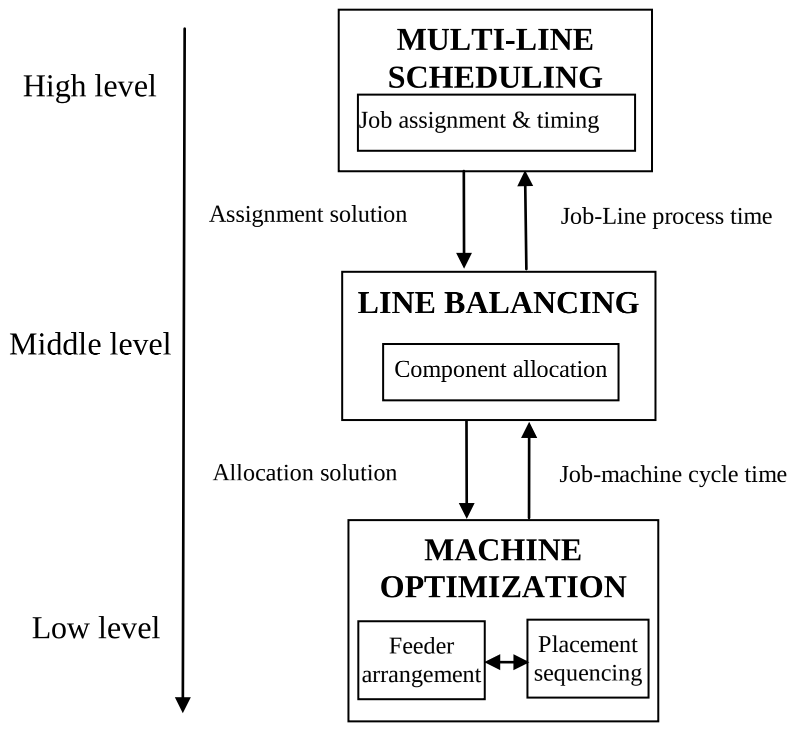


Figure 1.3 A general planning hierarchy in PCB assembly

First of all, PCB assembly orders (with each order being for a batch of PCB of a specific type) during the planning horizon are required to be assigned and scheduled on the assembly lines for production. Due to different configurations of the assembly lines, the process time for each PCB batch is dependent on the line it is assigned to. In addition, due dates of the jobs should be met to maintain customer satisfaction. Thus, the decision on this Multi-line Scheduling Problem (MLAP) should be made carefully in order to improve the production efficiency while meeting the due date requirement.

Second, after a batch of PCB is assigned to an assembly line, the Component Allocation Problem (CAP) is needed to be solved to determine which components on the board should be placed by which placement machine in the line (note that the placement process is always the bottleneck process in the line and there are more than one placement machine on each line). Similar to the traditional line balancing problem, the objective of the component allocation problem is usually to minimize the cycle time for the assembly line.

Finally, after the components are allocated to the placement machines in the line, the machine optimization problem is required to be solved for each placement machine according to the allocated placement task. Due to the unique characteristics of the placement process, there are two fundamental subproblems for machine optimization. The first is the Feeder Arrangement Problem (FAP), which determines the assignment of different component feeders to the feeder slots of the feeder carrier on the machine. The second is the Placement Sequencing Problem (PSP), which decides the sequence for the components to be placed onto the board. These two subproblems are important for reducing the assembly time for each single machine.

The above-mentioned problems are of great importance for the manufacturers to make effective use of most valuable resources (i.e., placement machines) and thereby offer opportunities for significant cost reductions. Increased production efficiency may reduce the need for additional capital expenditures for expensive equipment, and improve the production capacity at the same time.

1.2.2 A general planning process

Although the three planning problems in PCB assembly are of different levels at the planning hierarchy, they are highly interrelated (Figure 1.3). On one hand, the lower-level problems are based on the solutions to the higher-level problems. On the other hand, the solutions to the higher-level problems are influenced by the solutions to the lower-level problems.

For example, the machine optimization problems (i.e., the feeder arrangement and placement sequence) for each machine cannot be solved before component allocation decisions are made. At the same time, solutions to the component allocation problem cannot be evaluated without knowing the process time for each machine in the line, which in turn is determined by solving the machine optimization problems. Similarly, solutions to the component allocation problems provide the information on line cycle times for the scheduling problem.

Usually, the planning problems in PCB assembly are tackled in a decomposition manner. If in the planning period there are a set of n PCB jobs to be completed in an assembly shop with K assembly lines, the general planning process can be illustrated in Figure 1.4.

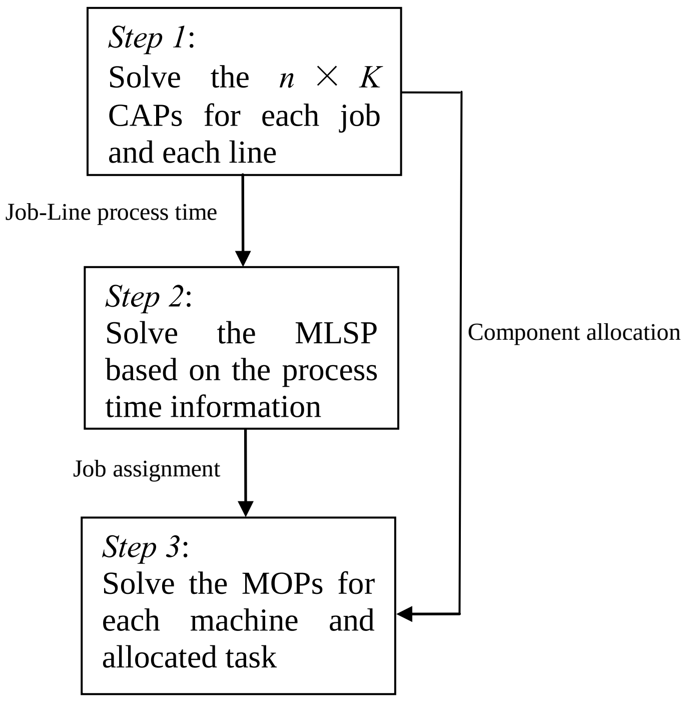

Step 1:
Solve the $n \times K$ CAPs for each job and each line
Job-Line process time
Step 2:
Solve the MLSP based on the process time information
Component allocation
Job assignment
Step 3:
Solve the MOPs for each machine and allocated task

Figure 1.4 A general planning process in PCB assembly

In the first step, a set of $n \times K$ Component Allocation Problems (with each corresponding to a job and an assembly line) are solved to obtain the cycle time information (including all the cycle time values for each job on each line). Based on the cycle time information, the process time for each job on each line can then be calculated as the product of the corresponding cycle time and the batch size. Sample results in the first step can be illustrated in Table 1.1.

In the second step, based on the process time information obtained in the first step, the Multi-Line Scheduling Problem can be solved to determine the assignment of the jobs to the assembly lines and the start time for each job. A sample results in this step can be illustrated in Table 1.2.

In the final step, based on the job assignment and corresponding component allocation decisions, the Machine Optimization Problems (MOPs) are solved to determine the feeder arrangement and placement sequence for each job on each machine.

Table 1.1 Process times for each job on each line (in hours)

	PCB 1	PCB 2	PCB 3	…	PCB *n*
Line 1	2.87	3.51	4.11	……	2.98
Line 2	5.60	4.84	6.34	……	4.87
Line 3	5.62	5.98	7.84	……	6.55
……	……	……	……	……	
Line *K*	4.54	3.97	4.27	……	5.60

Table 1.2 A solution to the multi-line scheduling problem

	PCB 1	PCB 2	PCB 3	…	PCB *n*
Assembly line	1	5	1	……	4
Start time (hour)	11.4	2.7	2.0	……	15.87

1.3 PROBLEMS

This thesis focuses on the first two planning problems in the planning hierarchy for PCB assembly, i.e., the Multi-Line Scheduling Problem (MLSP) and the Component Allocation Problem (CAP). The motivation of studying these two problems is discussed in the following.

First of all, because the machine optimization problems are machine-specific and the machine technologies evolve much rapidly, generally-applicable solution methods for these machine optimization problems is not realistic. Many machine vendors have provided optimization software to tackle these problems for their own machines, e.g., Flexa for Fuji machines, PT200 for Panasonic machines, UPS for Universal machines, HLC for JUKI machines, etc.

Comparatively, the CAP and the MLSP are of the higher planning levels at the planning hierarchy and do not depend on the machine technological characteristics as much as the machine optimization problems. Therefore, they are more general to different manufacturers. Hence, their significance has been recognized by many production managers. On the other hand, owing to the complexity and realistic constraints, current solution methods for the CAP and the MLSP are not effective and required to be investigated thoroughly.

For the component allocation problem, the solution is influenced by the machine optimization problems. The optimal solution to the component allocation problem can only be obtained through the integration of the problem with the machine optimization problems. However, due to the great computational complexity, it is impossible to solve the machine optimization problems whenever a component allocation solution is evaluated. For this reason, an effective and efficient solution method is required.

For the multi-line scheduling problem, production managers mainly concern about meeting the due dates and improving production efficiency at the same time. However, most researchers in the literature solve the job assignment problem without considering due date requirement. The resulting job assignment solutions are far from reality and the desired production efficiency can hardly be achieved. For this reason, the job assignment and timing of the jobs should be determined simultaneously. The formulation and modeling of the multi-line scheduling problem with due date constraints should be established and investigated. However, the due date constraints may greatly increase the complexity of the scheduling problem [Mok01]. There is a need to develop an efficient method to solve the problem.

1.4 OBJECTIVES

The ultimate aim of this research is to optimize the production plans for PCB assembly so that both production efficiency and customer satisfaction can be improved. To this end, two important planning problems are investigated, i.e., the Component Allocation Problem (CAP) and the Multi-Line Scheduling Problem (MLSP). Effective solution approaches for solving these problems are developed. There are five objectives to be achieved in this research, which are described as follows.

The first objective is to develop an effective decomposed solution strategy for solving the component allocation problem. Since the component allocation problem is combined with the machine optimization problems, in order to improve the efficiency for solving realistic-sized problems, the solution strategy will rely on a placement time estimator that can estimate the placement (process) time for a machine to complete a specific PCB. Based on the placement time estimation, an algorithm or heuristic could be developed to solve the component allocation problem effectively without tackling the machine optimization problems.

After the decomposed solution strategy has been proposed, the second objective is to examine the feasibility of developing the relative placement time estimator. A placement time estimator for a turret type placement machine will be constructed in this research. The placement time estimator should be able to estimate the placement time without solving the machine optimization problems. The effectiveness of the placement time estimator should be evaluated based on the statistical analysis.

The third objective is to develop a specific heuristic method for solving the component allocation problem, with the solutions being evaluated by the developed placement time estimator. The effectiveness and efficiency of the heuristic method should be examined through solving some problem instances.

Since the specific multi-line scheduling problem in PCB assembly has not been investigated in the literature, the fourth objective is to establish a complete mathematical model for the problem with an objective which considers both the production efficiency and due date satisfaction. The validation and the complexity of the models should be examined.

Since the multi-line scheduling problem is shown to be extremely complex, the fifth objective is to develop an effective and efficient heuristic method for solving the multi-line scheduling problem. The effectiveness and efficiency of the heuristic method should be examined through experimental tests.

1.5 SCOPE OF THIS THESIS

This research is mainly devoted to the two important planning problems in PCB assembly, i.e., the Multi-Line Scheduling Problem (MLSP) and the Component Allocation Problem (CAP). These problems arise from the environment relating to the high-mix and medium-to-high-volume production mode and the shop floors with multiple assembly lines. This manufacturing environment is commonly seen in many PCB manufacturing companies.

The structure of the thesis is organized as follows.

In Chapter 2, an extensive literature review is conducted to demonstrate what have been studied on the planning problems in PCB assembly, including the machine optimization problems, the component allocation problem, and the job assignment and scheduling problem. Since most of the planning problems in PCB assembly are difficult combinatorial problems, heuristic methods for solving combinatorial problems are also surveyed.

Chapter 3 is devoted to the component allocation problem. A decomposed solution strategy based on a regression-based placement time estimator is proposed for the problem. The purpose of this placement time estimator is to accurately and efficiently estimate the process time for each machine in the line without tackling machine optimization problems (that is, the feeder arrangement problem and the placement sequencing problems). A placement time estimator for a turret-type placement machine is developed and the effectiveness of the estimator is examined through some experimental tests. A genetic algorithm, which uses this placement time estimator for solution evaluation, is developed for solving the component allocation problem. Numerical results on some problem instances are reported.

In Chapter 4, the multi-line scheduling problem is investigated. A complete mathematical model for the multi-line scheduling problem is established. The model is verified through solving some randomly generated small-sized instances. Based on the established model, an effective and efficient genetic algorithm is developed. Numerical results on solving some problem instances and a case study are reported.

Finally, Chapter 5 summarizes the distinctive achievements of this research. Both the academic and industrial contributions of this research are concluded. Some recommendations for future work are suggested.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

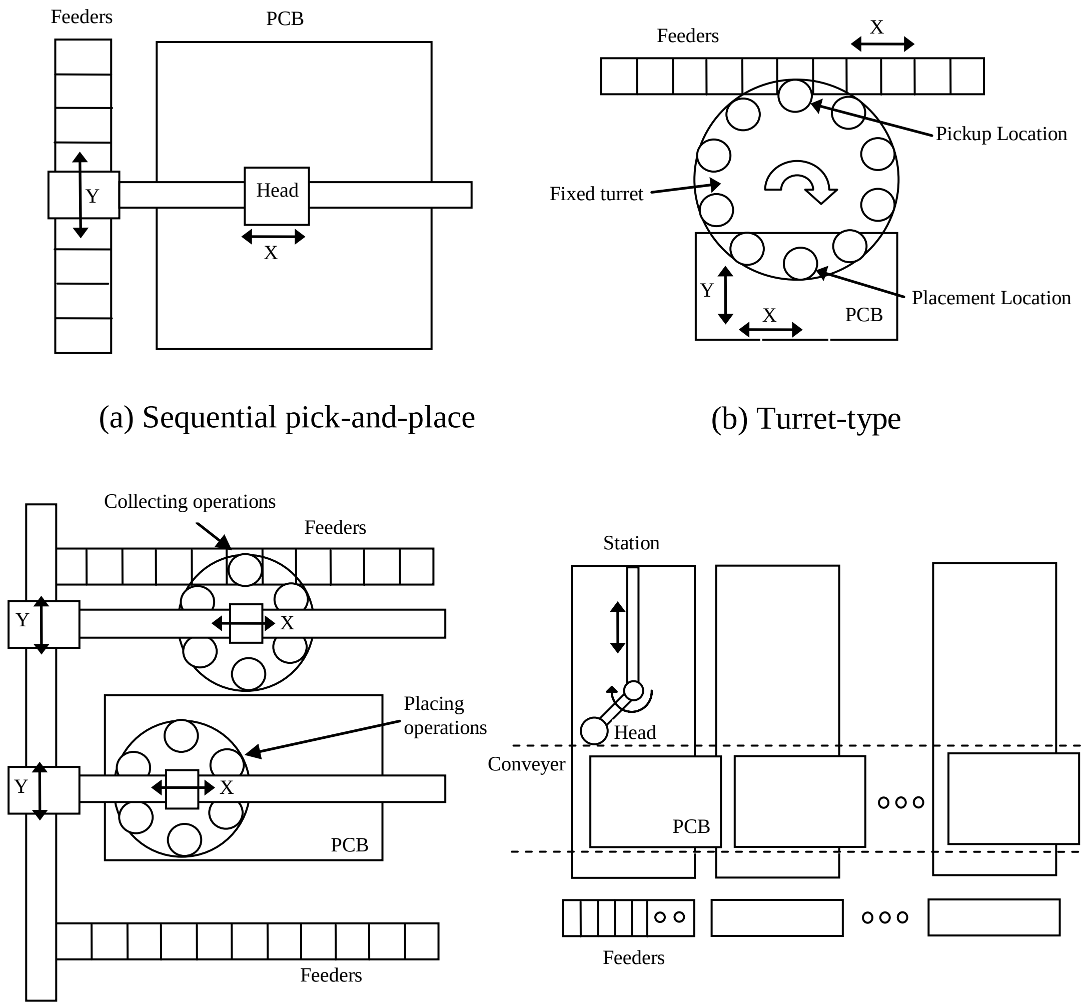
Numerous researchers have investigated the planning problems in PCB assembly. However, not all the problems have been studied intensively and thoroughly. In this chapter, a comprehensive literature review on these problems is conducted. The focus is to clarify what have been done and what have not, with the latter providing the rationale for this research.

Section 2.2 surveys relative research on the machine optimization problems, i.e., the problems of the lowest planning level. The main difficulty for solving these problems is discussed. After that, literature relating to the component allocation problem (CAP) is reviewed in Section 2.3. The main drawback of existing solution methods is discussed. In Section 2.4, literature relating to the line assignment and scheduling problem in PCB assembly is reviewed. Some realistic constraints for this problem, which are neglected in most of the research, are discussed. Since the investigated multi-line scheduling problem (MLSP) is found to be much similar to the well-investigated parallel-machine scheduling problem (PMSP), literature relating to the parallel-machine scheduling problem is also reviewed. In Section 2.5, a survey on the heuristic methods for combinatorial problems, which most of the planning problems in PCB assembly belong to, is presented. Finally, some remarks concerning the literature reviews are summarized in Section 2.6.

2.2 MACHINE OPTIMIZATION PROBLEMS

Electronic components (possibly hundreds or thousands) are assembled onto a board by PCB placement machines (Figure 2.1). A typical placement machine has a feeder carrier, a PCB table, and a placement device. The feeder carrier consists of many feeder slots where the component feeders are located. The component feeders are used to provide the machine with a continuous supply of components. Generally, the placement device picks components from the feeders and places them onto the board according to a certain operational method. Based on different operational methods, the machines can be classified into several categories: sequential pick-and-place [Bal88], turret-type [Leu93], single-gantry collect-and-place [Gru04], dual-gantry collect-and-place, multi-station [Csa00a], as shown in Figure 2.1.

Optimization of the feeder arrangement and sequencing of the component placements are two most important factors for improving the efficiency of placement machines. They correspond to two fundamental subproblems for machine optimization, i.e., the Feeder Arrangement Problem (FAP) and the Placement Sequencing Problem (PSP). The feeder arrangement problem is to decide the assignment of different component feeders to the slots on the feeder carrier, while the placement sequencing problem is to decide the placement sequence of the components.



(a) Sequential pick-and-place

(b) Turret-type

(c) Dual-gantry collect-and-place

(d) Multi-station

Figure 2.1 Different PCB placement machine types

Numerous researchers have conducted investigations into the machine optimization problems. Most researchers find that these problems are much difficult and intertwined with each other to influence the efficiency of the machine. Due to the great computational complexity, it is not realistic to use mathematical programming approaches to solve the combined problem of feeder arrangement and placement sequencing. Alternatively, there are three different ways in the literature to tackle the two problems in the literature. Some researchers focus on one of the two problems, assuming the solution to the other is given. Some researchers rely on an iterative procedure to tackle these two problems. The other researchers propose integrated heuristic or metaheuristic methods to solve the two problems simultaneously. The following survey is conducted according to this classification.

2.2.1 Separated solution methods

In the literature, some researchers focused on only one of the two machine optimization subproblems, i.e., the feeder arrangement problem and the placement sequencing problem, while assuming the solution to the other is given in advance.

Drezner and Nof [Dre84] are the first researchers who investigated the optimization problems for PCB placement machines. They considered a sequential pick-and-place machine and tackled the placement sequencing problem, assuming the feeder arrangement had been determined in advance.

Similarly, Ball and Magazine [Bal88] solved the placement sequencing problem, assuming the feeder arrangement was fixed. The problem was modeled as a rural postman problem and a heuristic approach was used to solve the problem.

De Souza and Wu [Des94] studied the placement sequencing problem for a turret-type machine. They incorporated a knowledge-based component placement system with a Traveling Salesman Problem (TSP) algorithm to solve the problem.

Kumar and Li [Kum95] established an integrated model for these two problems. The model was decomposed into a TSP and a minimum weight matching problem (MWMP) by adding extra constraints.

Moyer and Gupta [Moy96a] studied the feeder arrangement problem for the turret-type machine based on the assumption that the placement sequence was predetermined. The problem was formulated as the Quadratic Assignment Problem (QAP) and solved by two heuristic methods.

Ahmadi and Mamer [Ahm99] modeled the problem of sequencing the component types for placement and the problem of scheduling the movements between points on the PCB as a collection of interdependent TSPs.

Klomp *et al*. [Klo00] solved the component allocation problem for a turret-type machine and considered a feeder and its corresponding cluster (that is, a set of locations served by a single feeder) as a node in a complete graph.

Kim and Park [Kim04] also solved these two subproblems separately. A clustering algorithm and an assignment algorithm were applied to solve the feeder arrangement problem, while an assignment algorithm and a connection algorithm were applied to solve the placement sequencing problem.

Li *et al*. [Lis07] considered a collect-and-place machine with a revolving head. They first solved the placement sequencing problem as a TSP. Then the feeder arrangement problem was solved in the second stage using a Genetic Algorithm (GA).

2.2.2 Iterative solution methods

Some researchers focused on an approach which consists of tackling both problems (the feeder arrangement problem and the placement sequencing problem) by iterating their solutions.

Foulds and Hamacher [Fou93] used an iterative approach for a sequential pick-and-place machine. The placement sequencing problem was solved by a TSP heuristic and the feeder arrangement problem was solved by a QAP heuristic.

Egbelu *et al*. [Egb96] also considered a sequential pick-and-place machine and solved the two problems iteratively. The feeder arrangement problem was solved by a cutting plane and exchange heuristic, while the placement sequencing problem was solved by a composite procedure of a farthest insertion algorithm and a 3-opt local search.

Crama *et al*. [Cra90] adopted an iterative approach for a single-gantry collect-and-place machine. The authors solved the placement sequencing problem using the approach proposed by Leipälä and Nevalainen [Lei89] and solved the feeder arrangement problem using a simple local search method.

Grunow *et al* . [Gru04] considered a collect-and-place machine with a revolving head and solved the feeder arrangement problem and the placement sequencing problem iteratively. They first obtained initial feeder arrangement using a simple heuristic, and then solved the PSP as a simple Vehicle Routing Problem (VRP). After that, the initial solutions were improved by iteratively applying a 2-opt local search to one of the two solutions while fixing the other one.

2.2.3 Integrated solution methods

Many researchers proposed heuristic or metaheuristic methods for solving the feeder arrangement and placement sequencing problems simultaneously.

Leipälä and Nevalainen [Lei89] stated that the iterative approach was fast but obtained inferior solutions. They solved the feeder arrangement and placement sequencing problems integrally for a sequential pick-and-place machine. A simple pairwise exchange heuristic was used to solve the feeder arrangement problem, while the evaluation of each feeder arrangement solution was performed by solving the placement sequencing problem using a modified farthest insertion heuristic. A similar integrated approach was adopted by Sohn and Park [Soh96] for a turret-type machine.

Broad *et al*. [Bro96] established an integrated Integer Programming (IP) model for the two problems for a sequential pick-and-place machine. The model was solved by a binary integer programming package. They stated that realistic instances were solved efficiently using a 1% tolerance on the difference between the objective values from the integer solution and the Linear Programming (LP) relaxation.

Deo *et al*. [Deo02] also studied a sequential pick-and-place machine. They considered multiple setups which were necessary with limited feeder holding capacity. They proposed a genetic algorithm for tackling the integrated problem.

Leu *et al*. [Leu93] proposed a two-link genetic algorithm to simultaneously solve the feeder arrangement problem and the placement sequencing problem for both a sequential pick-and-place and a turret-type machine. Ong and Khoo [99] modified the two-link genetic algorithm proposed by Leu *et al*. [Leu93] and considered the case in which feeder duplication is allowed, i.e., components of the same type can be stored in more than one feeder. Ho and Ji [How03, How04] also modified the two-link genetic algorithm proposed by Leu *et al*. [Leu93] for both a sequential pick-and-place and a turret-type machine.

Moyer and Gupta [Moy96b] proposed a specific heuristic to solve the two problems simultaneously for a turret-type machine. The aim of the heuristic is to generate a placement sequence and feeder setup to exploit the unique characteristics of the turret-type machines. They argued that on average, their approach was superior to the genetic algorithm proposed by Leu *et al*. [Leu93].

Yeo *et al*. [Yeo96] proposed a rule-based approach to simultaneously solve the feeder arrangement problem and the placement sequencing problem for a turret-type machine. The approach is based on a one-pitch incremental feeder heuristic and a nearest neighbor heuristic.

Ellis *et al*. [Ell01] proposed a heuristic for solving the feeder arrangement problem and the placement sequencing problem simultaneously for a turret-type machine. They used a construction procedure with a set of rules to generate an initial component placement sequence and feeder arrangement, and an improvement procedure to improve the initial solution.

Magyar *et al*. [Mag99] studied a single-gantry collect-and-place machine. They proposed a hierarchical solution approach to solve the problem of determining the placement sequence, assignment of different nozzles to the robot head, and feeder setup.

Altimeter *et al* . [Alt00] proposed an integrated model for the feeder arrangement problem and the placement sequencing problem for a single-gantry collect-and-place machine, and devised an algorithm which converts these two problems into a number of vehicle routing problems.

Ayob and Kendall [Ayo05] also studied a single-gantry collect-and-place machine. They proposed a triple objective function to minimize the assembly time, feeder movements and PCB table movements.

For a dual-gantry collect-and-place machine, the efficiency is largely determined by the gantry workload and the gantry scheduling [Su05]. Tirpak *et al*. [Tir00] proposed an adaptive simulated annealing algorithm for solving three optimization problems simultaneously, i.e., the feeder setup, nozzle setup and placement sequencing.

Sun *et al*. [Sun05] also studied a dual-gantry collect-and-place machine. They proposed a genetic algorithm to decide the component allocation between the two revolving heads and feeder arrangement. In order to evaluate the workload of each delivery unit, they used a greedy heuristic for work cycle formation and pickup sequencing decisions. Computational performance was examined using real industrial data.

Kulak *et al* [Kul07] proposed genetic algorithms for both single-gantry and dual-gantry collect-and-place machines with revolving heads. The feeder arrangement problem and the placement sequencing problem were solved by genetic algorithms (GAs). A clustering algorithm was integrated in the GAs to group placement operations in each collect-and-place cycle.

Ho *et al*. [How07] adopted the genetic algorithm which was similar to that in [How03] to solve the feeder arrangement problem and the placement sequencing problem simultaneously for a dual-gantry collect-and-place machine with revolving heads.

Compared with other types of placement machine, the optimization of multi-station machines has been tackled by relatively few researchers. Due to concurrency of the stations, synchronization is the most crucial factor for the optimization problem [Csa00a].

Wang *et al*. [Wan99] proposed a genetic algorithm to optimize feeder setup for a multi-station placement machine. Crasser *et al* . [Csa00b] employed a knowledge-based system to optimize a multi-station machine. The system divides the optimization problem into two subproblems, i.e., assignment of components to the stations, and feeder arrangement within the stations.

In other work, Crasser *et al*. [Csa00a] also studied a multi-station placement machine and proposed a two-phrase approach. Since the machine has many stations, they tackled component allocation to stations, feeder setups and placement sequencing for each station. They partitioned the optimization problem into two phases and solved them using a tabu search and a specific heuristic, respectively.

Recently, Grunow *et al*. [Gru03] established an integer programming model for the optimization of a multi-station machine. They proposed two different solution procedures for the problem with the aim of balancing workloads of stations.

It can be seen from the above survey that numerous researchers have investigated the machine optimization problems. However, solutions to the machine optimization problems are not general for all types of machines. All the researchers only proposed a specific solution method for the machine they investigated. Due to the ever-advancing technology and the large variety of placement machines, development of a general solution is far from feasible.

On the other hand, there seems to be a gap between the research in the literature and industrial solutions. In fact, different machine vendors have provided optimization software to tackle the machine optimization problems for their own machines. For example, there are Flexa for Fuji machines, PT200 for Panasonic machines, UPS for Universal machines, HLC for JUKI machines, etc. Most of the software provides effective and efficient solutions to machine optimization. However, few researchers in the literature have conducted comparison between their solutions with existing solutions in the industry.

2.3 THE COMPONENT ALLOCATION PROBLEM

The component allocation problem (CAP) arises when a batch of PCB is assigned to an assembly line with multiple placement machines for production. The CAP is to allocate the components required by the PCB to machines in the line so that the makespan or cycle time is minimized.

Compared with the machine optimization problems, research on the component allocation problem has received less attention.

As discussed in Chapter 1, the solution to the component allocation problem is tightly intertwined with the solutions to the machine optimization problems because the latter determine the actual process time for each machine (see Figure 1.3). Most researchers tackled the component allocation problem individually by using oversimplified estimation methods for obtaining process time, which reduce the effectiveness of the solutions.

2.3.1 Approaches based on roughly estimated process times

Crama *et al* . [Cra90] proposed a hierarchical approach to solve the component allocation problem for the collect-and-place machines, assuming constant unit time for a machine to place the components of the same type. In this way, the component allocation problem was solved separately without considering the machine optimization problems. They argued that the approach obtained results similar to those obtained by the approach proposed in a confidential report of the Philips Center for Quantitative Methods. However, they agreed that the process time for a component is actually not constant and dependent on the solutions to the machine optimization problems. Therefore, the solution obtained in this way cannot be effective.

Ji *et al* . [Jip01] established a mixed integer programming model for the component allocation problem. The model also uses estimated unit time for processing a component of a specific type. In their model, components of the same type are allowed to be allocated to different machines, that is, feeder duplication is allowed. They proposed a genetic algorithm to solve the problem which yields less than 1% difference between the best found solutions and optimal solutions. However, the solutions are also evaluated based on the roughly estimated process time values.

Kodek and Krisper [Kod04] addressed the component allocation problem and proposed an optimal branch-and-bound-based algorithm. They showed that the algorithm could obtain the optimal solution for the problems with up to 50-80 variables. However, they also assumed the constant unit time for components of a specific type.

Some researchers considered the case for multiple boards to be produced on the same line without feeder changeover. Ben-Arieh and Dror [Ben90] proposed simple heuristics for a case with two machines and multiple boards. They stated that the solution results were within 0.5% from optimality. However, as mentioned in the paper, they also assumed that the process time was one unit for each component, regardless of the board type and component location on the board.

McGinnis *et al*. [Mcg92] stated that the most appropriate objective function for the multiple-board case consists of minimizing the sum over all board types of the process time of these board types on their bottleneck machines. Similar to those in the aforementioned papers, the model they proposed uses estimated placement time p_{jkm} for machine m of all components of type j on board k. The authors mentioned that these placement time values must be roughly estimated, since feeder allocation, feeder location and placement sequencing decisions eventually interfere with each other to determine actual assembly time.

Ammons *et al* . [Amm97] also considered the multiple-board case and proposed a more general feeder allocation model by allowing for feeder duplication and partial feeder setups. They solved the mixed integer programming model by branch-and-bound. In the model, they also used rough estimates of placement time for a component of a specific type. They mentioned that these estimates yielded a poor approximation of actual makespan. An industrial case study was presented.

Throughput improvements of up to 8-10% over the company's current manual procedure conducted by the process engineer were obtained. DePuy *et al*. [Dep01] proposed an integer programming heuristic for the same problem and conducted several case studies to demonstrate the effectiveness of the method.

With considering sequence-dependent setup times (feeder changeover), Gronalt and Zeller [Gro00] investigated both the component allocation problem and the job sequencing problem to minimize makespan for an assembly line with two placement machines to produce boards of multiple types. Two heuristic procedures were proposed and proved to be effective for solving some real problems.

Ashayeri and Selen [Ash07] also considered the feeder changeover time and solved the job sequencing problem along with the component allocation problem for each job. The authors proposed two decomposed planning strategies: one focuses on minimum number of changeovers and the other on minimum process time. They stated that both strategies did not deviate excessively from optimal solutions.

2.3.2 Approaches based on actual process times

All the models in above-mentioned research assume constant process time for a specific machine to place a component of a specific type. However, the actual placement times may differ a lot from these rough estimates. Some influential factors should be considered, like the closeness of the components, number of feeders required. These factors are key factors considered in the machine optimization problems, i.e., the feeder arrangement problem and the placement sequencing problem, which eventually determine the process time for a machine.

Only a few researchers have considered the interaction between the component allocation problem and the machine optimization problems. Lapierre *et al*. [Lap00] tackled the feeder allocation problem for a line and the feeder arrangement problem for each machine simultaneously. They claimed that, for the particular machine type considered in their paper, the placement time is independent of the placement sequence. Therefore, they assumed that the process time of a component was constant when feeder arrangement was determined. In this way, they were able to establish a model which simultaneously determines feeder allocation to machines and feeder arrangement on each machine. They proposed Lagrangian relaxation techniques to solve the integrated problem and stated that the techniques obtained little difference between the best found solutions and the lower bounds. However, Duman [Dum05] recently proved that the assumption made by Lapierre *et al*. is not true and showed that placement time is actually dependent on placement sequence even for the concerned machine.

Crama *et al*. [Cra97] proposed a heuristic method to estimate the process time for a machine to place all allocated components. However, the heuristic exploits the operation characteristics of the turret-type machines and cannot be easily applied to other machine types. In addition, the proposed estimation method requires nontrivial computational efforts. As the result, the authors can only use a simple local search to solve the component allocation problem.

2.4 THE JOB ASSIGNMENT AND SCHEDULING PROBLEM

The scheduling problem investigated in this thesis arises when a set of PCB batches are required to be processed by multiple assembly lines. In this research, the problem is named as the Multi-Line Scheduling Problem (MLSP). The problem is to simultaneously determine the assignment of jobs to assembly lines and the sequencing of jobs in each line, so that the makespan is minimized while meeting due date requirement. This section surveys the literature relating to the line assignment and scheduling problem in PCB assembly. Because the Multi-Line Scheduling Problem (MLSP) is found to be much similar to the Parallel-Machine Scheduling Problem (PMSP), literature relating to the PMSP is also discussed.

2.4.1 The line assignment problem in PCB assembly

In the literature, most researchers mainly focused on solving the line assignment problem without considering the due dates of the jobs. The line assignment problem is aimed to improve the production efficiency through appropriate assignment of jobs to lines. Even without considering the due date requirement, the line assignment problem is complex due to the job and line dependent process times.

Balakrishnan and Vanderbeck [Bal99] considered the line assignment problem in a high-mix, low-volume environment. With the use of a partial setup policy, the objective of the problem is to minimize the setup cost while adding an upper-bound on the allowed workload per line. They established an integer programming model for the problem and proposed a heuristic method based on column generation.

Ellis and Bhoja [Ell02] considered the line assignment problem together with the component allocation problem. They solved the two problems in a decomposed manner. They stated that by solving the component allocation problem the cycle time for each line can be obtained. For the line assignment problem, they considered both the objective of minimizing the total line time (total production time for all the lines) and the objective of balancing the workload across the lines. They formulated the problem as a mixed-integer programming (MIP) model and solved it using an MIP solver with certain optimality tolerance.

Ji and Ho [Jip05] also addressed the line assignment problem with the objective of minimizing the total production time. A PCB batch is allowed to be split and produced on different lines. Similar to the model proposed by Balakrishnan and Vanderbeck [Bal99], an upper-bound is set for the allowed workload per line. They stated that assignment of a small quantity of boards to a line was impractical and should be associated with penalty in the objective function. They proposed a genetic algorithm which was shown to be able to solve the problem effectively and efficiently.

Çatay *et al*. [Çat06] considered the job assignment problem in an open shop environment in which the machines were decoupled and each PCB job could have more than one operation to be processed by different machines. They proposed a three step hierarchical scheduling methodology for solving the problem. However, as stated by the authors, the flow-line organization of machines, as investigated in this thesis, is more typical for most PCB manufacturers.

Feo and Bard [Feo95] seem to be the only researchers who consider due date constraints in the scheduling problem for PCB assembly. They developed a greedy constructive heuristic for solving the problem. In the heuristic, a composite of slack time and process time are used for the greedy function during the solution construction. The users are allowed to rank the schedules obtained in multiple runs of the heuristic according to their own objectives, e.g., weighted throughput and weighted tardiness.

Comparatively, the multi-line scheduling problem (MLSP) investigated in this research involves some practical constraints and considerations that have been neglected in related research in the literature, which can be described as follows.

First, most researchers do not consider the due dates of the jobs. A realistic objective of the scheduling problem should not only consider improving production efficiency, but also due date requirement. In addition, each job may have its ready time due to reasons like material availability. For most PCB manufacturers, these time constraints are practical. Without considering the ready times and due dates in the scheduling model, the desired efficiency cannot be achieved in real production. On the other hand, these time constraints may greatly increase the complexity of the scheduling problem [Mod01] and necessitate an efficient solution method.

Second, the setup (or transition) time for a job on an assembly line may depend on the job that is previously processed on the machine. In an investigated PCB manufacturer, for example, the PCBs are classified into two types: those meet the RoHS (Restrict of Hazardous Substances) compliance and those do not. The setup time for an RoHS job that immediately follows a non-RoHS job is longer than the setup time otherwise. Therefore, the setup time can be seen as sequence dependent, which adds extra complexity to the problem.

Third, there may be precedence requirements between the jobs. For example, it is common for both sides of a PCB to be processed. Precedence constraints exist for the two jobs, which process each side of the same board.

All the above considerations are realistic for many PCB manufacturers and should be considered in the scheduling problem. However, to the best of our knowledge, such a specific multi-line scheduling problem in PCB assembly has not been investigated in the literature.

2.4.2 The parallel-machine scheduling problem

Based on the investigation on other production scheduling problems in the literature, it is found that the Multi-Line Scheduling Problem (MLSP) investigated in this research is much akin to the Parallel-Machine Scheduling Problem (PMSP), if each assembly line in the MLSP is considered as a single machine.

The parallel-machine scheduling problem has been intensively studied in the literature. A survey on the literature relating to the parallel-machine scheduling problems is presented in this section, followed by the discussion on the gap between the multi-line scheduling problem in PCB assembly and the parallel-machine scheduling problems which have been studied in the literature.

The Parallel-Machine Scheduling Problem (PMSP) considers scheduling a set of jobs J_i (i = 1, …, n) on a set of parallel machines M_j (j = 1, …, m) to optimize a certain performance measure [Mok01]. Each job can be completed by any one of the m machines.

There are many performance criteria to be considered when solving the parallel-machine scheduling problem. The criteria can be classified into two types, i.e., the criteria based upon completion time measures, and the criteria based upon due date measures, while the latter greatly increases the complexity of the scheduling problem [Mok01].

Based on the machines, the PMSP can be divided into three types, i.e., identical, uniform, and unrelated [Che90]. For the identical machines, the process time for a job is independent of the machine which processes it; for the uniform machines, each machine has a different speed, s_j (j = 1, …, m) and the process time for a job is the basic job process time divided by the machine speed, $p_{ij} = p_i/s_j$; for unrelated machines, there is no particular relationship among process time values for the jobs and thus a matrix p_{ij} (i = 1, …, n; j = 1, …, m) for the process times is needed.

For the multi-line scheduling problem in PCB assembly, the process time for a job is dependent on the assigned line and there is no particular relationship among the process time values for the jobs, so the multi-line scheduling problem is more similar to the unrelated parallel-machine scheduling problem.

Sotskov and Shaklevich [Sot95] stated that the identical parallel-machine scheduling problem with makespan minimization, which is a relatively easy type of

PMSP, is *NP-hard*. It is unlikely that polynomial algorithms may exist to solve the problem unless $P = NP$ [Coo71, Kar72].

Exact algorithms are available mainly for the identical PMSP, for example, the branch and bound algorithms [Del95; Elm74; Bar77; Sar88; Bel94; Yal02; Nes08; Shi08], and the dynamic programming [Rot66; Law69; Gra79; Len80; Leu82]. However, as mentioned by Mokotoff [Mok01], these enumerative algorithms can only solve some small-sized problem instances. Furthermore, the unrelated parallel-machine scheduling problems are much more difficult than identical or uniform machine scheduling problems [Mar97].

Recently, Bard and Rojanasoonthon [Bar06] developed a branch-and-price algorithm for unrelated PMSP and successfully solved many 100-job instances. However, their problem has an uncommon objective which is to maximize the weighted number of jobs scheduled, where a job in a higher priority class has more weight or value than a job in a lower priority class.

Most researchers focus on developing heuristic methods for solving the parallel-machine scheduling problems. Hübscher and Glover [Hüb94] presented a tabu search for the identical PMSP to minimize the makespan. They introduced an influential diversification which improves the behavior and quality of the solutions obtained by the general tabu search. França *et al*. [Fra94] proposed a composite heuristic and stated that the heuristic could achieve near-optimal solutions to the testing instances in short computational times. Scutella *et al*. [Scu00] considered the same problem by introducing new local search techniques whose neighborhood structure is based on multiple exchanges of jobs among machines. They showed that, by means of the proposed algorithms, near optimal solutions could be obtained when the running time was not important and satisfactory ones could be found rapidly.

For the unrelated PMSP to minimize the makespan, van de Velde [Van93] presented a heuristic based on an iterative local search. Glass *et al*. [Gla94] and Piersma and van Dijk [Pie96] presented heuristics using local search which were shown to be more efficient. Glass *et al*. [Gla94] compared the relative performance of Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS) on the same problem. They stated that for these three algorithms, TS generates slightly better solutions in a short time, and GA and SA improves as the time limit increases. Piersma and van Dijk [Pie96] developed a new local search algorithm for a similar problem. They showed that a tabu search with an efficient neighborhood search strategy performed better than general local search algorithms. Srivastava [Sri98] also presented a tabu search which was shown to be very effective to provide near optimal solutions. Sourd [Sou99] presented two algorithms based on large neighborhood improvement procedures. One is based upon a partial and heuristic exploration of a search tree, and the other one is based on the duality approach of van de Velde [Van93]. Frangioni *et al*. [Fra04] proposed new local search algorithms for the same problem. They stated that the new approaches have better performance than the branch-and-bound algorithm proposed by Dell'Amico and Martello [Del95] and the heuristic proposed by França *et al*. [Fra94]. Rojanasoonthon and Bard [Roj05] considered the unrelated PMSP to minimize the weighted number of scheduled jobs, where a job in a higher priority class has more value than a job in a lower priority class.

As the time constraints like due date constraints are presented, the complexity of the scheduling problem will be greatly increased [Mok01]. Bean [Bea94] developed a genetic algorithm for the identical multiple machine problem to minimize the total tardiness, introducing random keys representation scheme to maintain feasibility from parent to offspring. Sivrikaya-Serifoglu and Ulusoy [Siv99] provided two genetic algorithms for a more complex problem that appears when scheduling a set of independent jobs, with different due dates and ready times, sequence-dependent setups, on a set of identical parallel machines with the objective of minimizing the sum of weighted earliness and tardiness. Suresh and Chaudhuri [Sur96] proposed a tabu search algorithm considering a bicriteria objective to minimize the makespan and maximum tardiness. For the scheduling problem to minimize the maximum lateness on unrelated machines, a tabu search based on the adaptive memory search was presented by Smutnicki [Smu98]. Armentano and de França Filho [Arm07] considered the uniform PMSP to minimize the total tardiness and proposed an adaptive memory-based GRASP (Greedy Randomized Adaptive Search Procedure) approach for the problem. Anghinolfi and Paolucci [Ang07] investigated the uniform PMSP with ready times and sequence-dependent setup times to minimize the total tardiness. They proposed a hybrid metaheuristic approach which integrates several features from tabu search (TS), simulated annealing (SA) and variable neighbourhood search (VNS). Zhou *et al*. [Zho07] proposed a new ant colony approach for solving the unrelated PMSP to minimize the total weighted tardiness and compared the results with a general ant colony algorithm.

As mentioned earlier, the Multi-Line Scheduling Problem (MLSP) in PCB assembly can be seen as a special type of unrelated parallel-machine scheduling problem (PMSP) with sequence-dependent setup times and precedence constraints. The objective of the MLSP considers both the production efficiency and due date requirement. The unique objective and additional constraints add extra complexity to the *NP-hard* unrelated PMSP. Both a complete mathematical model and an efficient solution method are required for the investigated multi-line scheduling problem.

2.5 HEURISTIC METHODS

Most of the optimization problems in PCB assembly are difficult combinatorial optimization problems. By allowing enough time, an exact algorithm can produce an optimal solution for a combinatorial optimization problem of small size. However, these exact algorithms are usually inefficient due to the time they require. In practice, a heuristic solution is highly desirable. In this section, a survey is conducted on the heuristic methods for combinatorial optimization. The heuristic methods that have been used for solving the planning problems in PCB assembly in the literature are also discussed.

2.5.1 Heuristic methods for combinatorial optimization

Many surveys of heuristic approaches such as [Sil80] and [Zan89] attempt to classify the heuristics into several broad categories: construction methods (usually for generating initial solutions), neighborhood search (improvement) techniques, relaxation techniques, etc. The other common classification of heuristics is single solution approaches and population-based approaches [Blu01]. Basic local search (deterministic iterative improvement), simulated annealing, tabu search etc., are examples of single solution approaches, whereas genetic algorithms, ant colony algorithms, evolutionary strategies, etc., are examples of population-based approaches.

A constructive heuristic (also known as a greedy approach) constructs a solution based on some criteria. The aim of a constructive heuristic is to build a solution from scratch. Some of the common constructive heuristics are nearest neighbor, multiple fragment and insertion heuristics [Joh90]. These approaches are often simple but practical as an initialization method that can produce an initial solution for starting the local search. Many constructive heuristics are problem-specific in order to satisfy the problem constraints.

Comparatively, a neighborhood search is more general. It attempts to improve the solution by exploring the neighborhood of the present solution [Aar97]. The neighborhood of a solution is the set of solutions that are close to the current solution in some sense. The important decision in a neighborhood search is of deciding the neighborhood structure and how to explore solutions in the neighborhood. A basic neighborhood search begins the search from a given solution, and then iteratively tries to improve the solution quality by applying move operators. The search stops when it gets trapped in a local optimum or the stopping criteria are met.

Traditional neighborhood search methods can generate good solutions efficiently at a reasonable computational cost but without being able to guarantee optimality [Ree95]. More advanced heuristic approaches, called metaheuristics, guide local search heuristics to escape from local optima [Ree95]. Some of the common metaheuristics are Tabu Search (TS), Simulated Annealing (SA), and Genetic Algorithms (GA).

Tabu search (TS), primarily suggested by Glover and Hansen [Glo89, Glo90], makes use of memory structures and incorporates the deterministic improvement algorithm (i.e. the descent method) with the possibility of accepting a worse solution in order to escape from local optima [Aar97]. It is a systematic search approach that exploits adaptive memory structures [Glo90]. The best legal neighbor of the current solution is always selected even if that solution is worse than that of the current solution. To prevent a cyclic move (moving back to a recently visited solution), the set of legal neighbors is restricted by a tabu list. However, an illegal neighbor that attains a certain aspiration level can still be accepted. The flowchart of a standard TS method is illustrated in Figure 2.2 [Glo93].

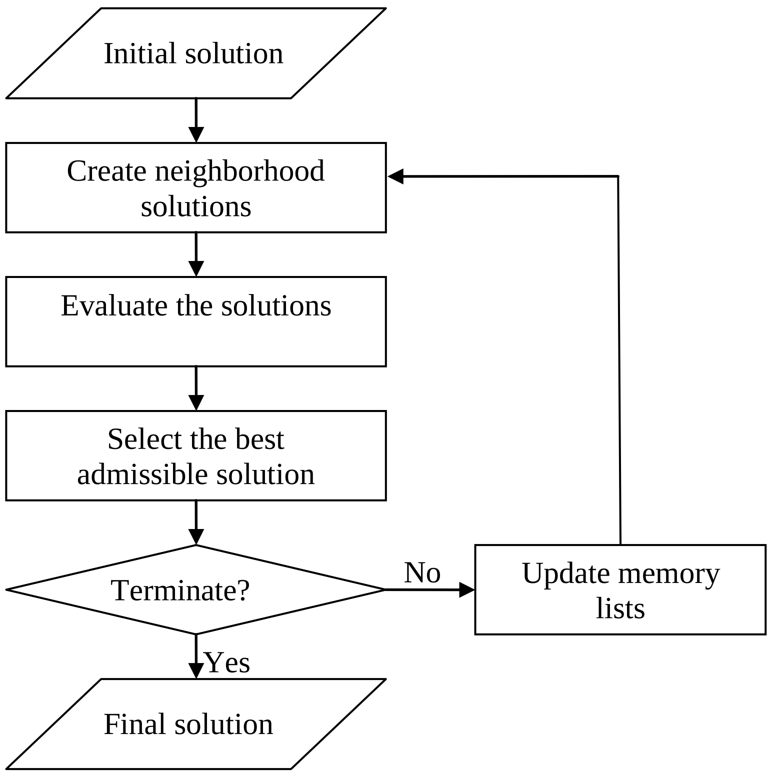


Figure 2.2 Flowchart of a standard tabu search method

Simulated annealing (SA), first proposed by Kirkpatrick *et al*. in 1983 [Kir83], is motivated from the analogy between combinatorial optimization problems and the physical annealing of solids (crystals) [Aar89] in which a solid is heated until it melts and is then slowly cooled to a state of minimum energy such that a uniform crystal structure, that is said to be in ground state, can be developed. The analogy associates the states of the physical system with the set of solutions, the physical energy of the solid as the objective function while the ground state is a globally optimal solution. The main idea of simulated annealing is to accept all improving solutions while probabilistically accepting worse solutions based on a control parameter (i.e. temperature in physical annealing). A cooling schedule is a vital component of the simulated annealing algorithm. It includes the upper and lower limit of the temperature parameter and the rate at which the temperature is reduced. The algorithm begins with a high temperature, which means a high probability of accepting worse solutions. As the search progresses, the temperature is gradually decreased, as such reducing the probability of accepting non-improving solutions. At temperature zero, the algorithm only accepts improving solutions. The algorithm ends when a stopping condition is met. Figure 2.3 summarizes the general SA procedure [Pha00].

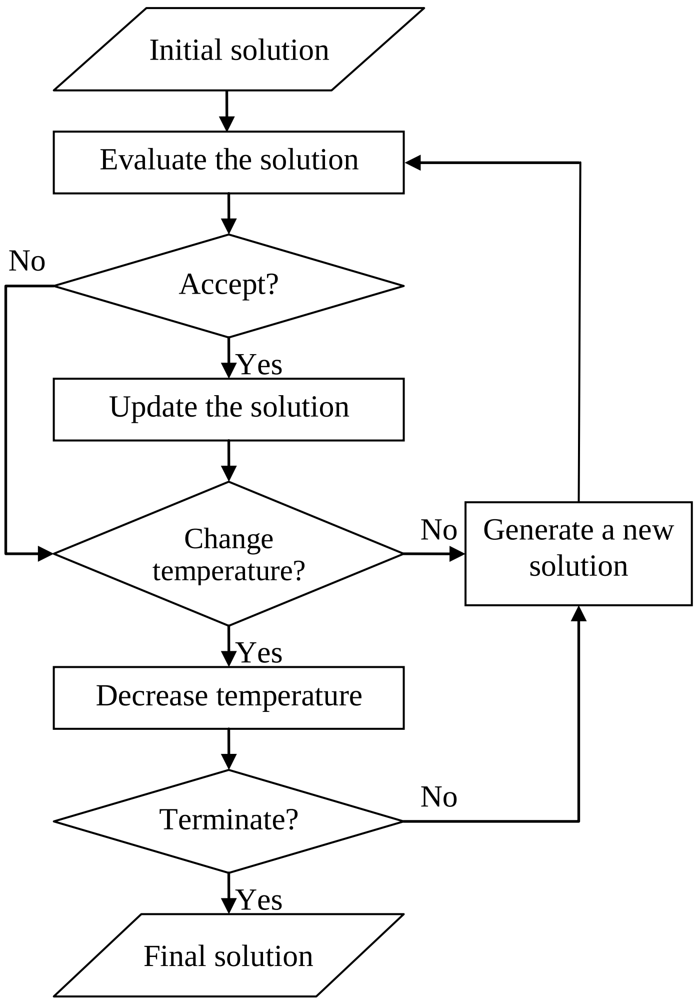


Figure 2.3 Flowchart of a standard simulated annealing method

Genetic algorithms (GA), originally developed by Holland in the 1960's, are a population-based method inspired by the principles of natural evolution [Man99]. The algorithm starts the search with a population of individual chromosomes (solutions) generated randomly or heuristically. In each generation (iteration), the population is evolved using genetic operators such as mutation and crossover to produce offspring (new individuals of the next generation). Mutation is a 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 usually chooses the parents based on survival of the fittest individuals. That is, the better fitness values are more likely to be chosen to undergo reproduction in order to produce offspring. The whole procedure is shown in Figure 2.4 [Dav91].

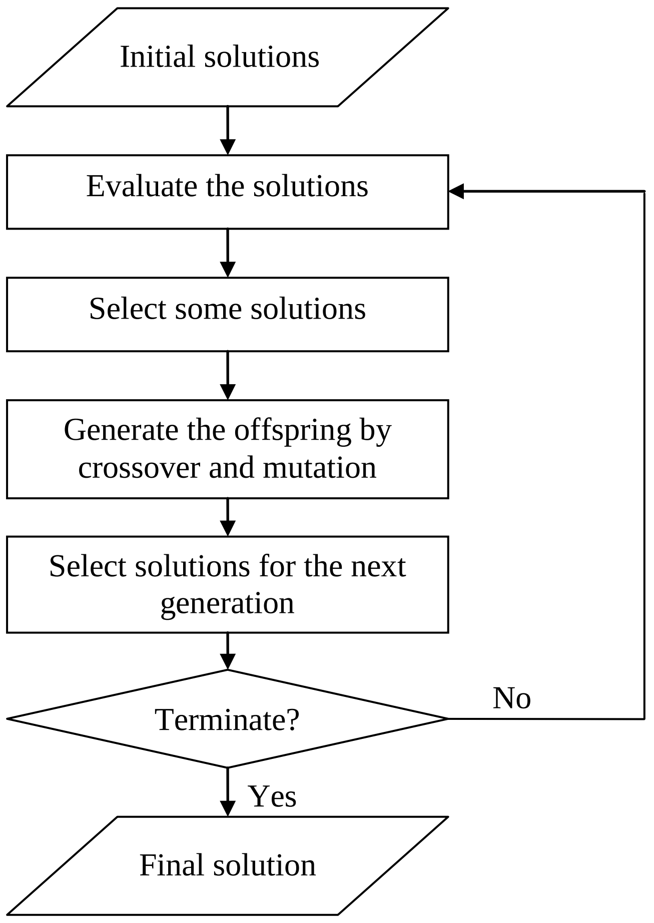


Figure 2.4 Flowchart of standard genetic algorithms

2.5.2 Heuristic methods used in PCB assembly

Because most of the planning problems in PCB assembly are difficult combinatorial optimization problems, most researchers rely on heuristic methods for solving these problems.

For the machine optimization problems which are *NP*-hard, many researchers proposed specific heuristic methods. Ball and Magazine [Bal88] proposed a heuristic approach for solving the placement sequencing problem. Moyer and Gupta [Moy96a] tackled the feeder arrangement problem by using two heuristic methods. For the more difficult combined problem of feeder arrangement and placement sequencing, many researchers also relied on heuristic methods [Cra90; Gru04; Moy96b; Yeo96; Ellis01].

Besides traditional heuristic methods, many researchers proposed metaheuristics for solving the machine optimization problems. Among these metaheuristic methods, genetic algorithms have been used extensively and demonstrated successful applications. For example, Khoo and Loh [Kho00] developed a prototype genetic algorithm to solve the machine optimization problems for a Fuji FCP-IV machine. Wang *et al*. [Wan99] argued that their genetic algorithm method performed as well as a human expert in optimizing the feeder arrangement problem for a Fuji QP-122 machine. Several researchers used genetic algorithms to solve the integrated problem of feeder arrangement and placement sequencing and achieved good results [Leu93; Ong99; Deo02; HoW03; HoW04]. Sun *et al*. [Sun05], Kulak *et al* [Kul07], and Ho *et al*. [How07] also relied on genetic algorithms for solving the optimization problems for collect-and-place machines. Besides genetic algorithms, simulated annealing was used by Tirpak *et al*. [Tir00] to solve the optimization problems for a dual-gantry collect-and-place machine. Csaszar *et al*. [Csa00a] proposed a tabu search and a specific heuristic to optimize a multi-station placement machine.

Heuristic methods have also been used to solve the component allocation problem [Cra97; Ben90; Gro00; Lap00] and the line assignment problem [Bal99; Feo95]. Nevertheless, genetic algorithms have also demonstrated successful applications for these high-level problems. Ji *et al* . [JiP01] proposed a genetic algorithm to solve the component allocation problem which obtained less than 1% difference between the best found solutions and the optimal solutions. For the line assignment problem, Ji and Ho [Jip05] developed a genetic algorithm and obtained good results. They stated that the proposed genetic algorithm could solve the problem both effectively and efficiently.

2.6 SUMMARY

In this chapter, an extensive literature review on the planning problems in PCB assembly, i.e., the machine optimization problems, the component allocation problem, and the line assignment and scheduling problem, has been conducted. As all these problems are much difficult combinatorial optimization problems, literature relating to heuristic methods for combinatorial optimization problems has also been reviewed. Some remarks concerning the reviews can be summarized as follows.

1. The machine optimization problems have been investigated intensively. Most researchers rely on heuristic methods for solving these problems due to the great complexity of the problems. Nevertheless, a generally applicable solution approach is not available because of the large variety of machine types and ever-advancing technologies. Most of the machine vendors have provided optimization software for their own machines, which exploits the technological characteristics of their machines.
2. The component allocation problem (CAP) has also been investigated by quite a few researchers. However, no effective approach has been proposed to

consider the interaction between the component allocation problem and the lower-level machine optimization problems. The objective values for the component allocation problem algorithms proposed in the literature are mainly based on the over-simplified estimate of process time for each machine. Therefore, the solutions obtained by the current CAP algorithms are not good enough due to the estimation error.

3. The line assignment and scheduling problem in PCB assembly has received relatively few attentions in the literature. Most of the relative research focuses on the line assignment problem for improving the production efficiency, while neglecting some realistic constraints like the ready time and due date for each job. These constraints may significantly increase the computational complexity of the problem. An efficient heuristic method is required.
4. The multi-line scheduling problem (MLSP) in PCB assembly investigated in this research is found to be much similar to the well-known unrelated parallel-machine scheduling problem (PMSP). However, the sequence-dependent setup times, the job precedence constraints, and the unique objective which considers both production efficiency and due date requirement, may greatly increase the complexity of the problem. A complete mathematical model and an efficient solution method for the specific MLSP have not been established in the literature.
5. Due to the simplicity and flexibility, genetic algorithms have been used extensively and demonstrated successful applications in solving the planning problems in PCB assembly. In this project, the genetic algorithms will also be

proposed to solve the investigated planning problems, i.e., the component allocation problem (CAP), and the multi-line scheduling problem (MLSP).

In the next chapter, a detailed description of the component allocation problem (CAP) will be given. An effective solution strategy will be proposed. Experimental tests on some problem instances will be conducted to examine the effectiveness and efficiency of the proposed method.

CHAPTER 3

THE COMPONENT ALLOCATION PROBLEM (CAP)

3.1 INTRODUCTION

When a batch of PCB is assigned to an assembly line for processing, the Component Allocation Problem (CAP) is required to be solved to allocate components required by the PCB to the placement machines in the line, with the objective of minimizing the makespan or cycle time.

As discussed in Section 1.2, the component allocation problem is dependent on the solutions to the lower-level machine optimization problems, which eventually determine the actual process time for each machine. In this chapter, a decomposed solution strategy is proposed for solving the Component Allocation Problem (CAP). The solution strategy relies on a so-called placement time estimator, which can accurately estimate the placement time for a machine without solving the machine optimization problems. Based on the placement time estimator, an algorithm or heuristic can be developed to solve the component allocation problem effectively without tackling the machine optimization problems.

The structure of this chapter is organized as follows. The component allocation problem is formulated and described in detail in Section 3.2. The decomposed solution strategy is proposed and described in detail in Section 3.3. The development of a placement time estimator is described in Section 3.4, followed by the development of a specific genetic algorithm for solving the component allocation problem in Section 3.5. Numerical results on solving some problem instances are presented in Section 3.6. The main work of this chapter is summarized in Section 3.7.

3.2 PROBLEM FORMULATION

If a batch of a specific PCB is assigned to an assembly line with multiple placement machines for production, one should decide the allocation of the components required by the PCB to the placement machines in the line so that the process time on the bottleneck machine is minimized. In the literature, this problem is called the Component Allocation Problem (CAP).

In practice, the components of the same type may be split and assigned to more than one machine in the line, which is called feeder duplication. However, some manufacturers prefer not adopting feeder duplication due to the expensiveness of the feeders. Feeder duplication is not considered in this research. Without feeder duplication, the component allocation problem is reduced to allocating the component feeders (with each feeder supplying components of a specific type) to the machines.

Figure 3.1 shows an example for the component allocation problem. In the example, a batch of PCB with 22 components of 8 different types is to be assembled by an assembly line with 4 placement machines. Each machine in the line is responsible for placing components of 2 types, so that the greatest process time among the 4 machines is minimized. For a realistic PCB, there may be hundreds of components and dozens of component types, and the component allocation problem is very complicated.

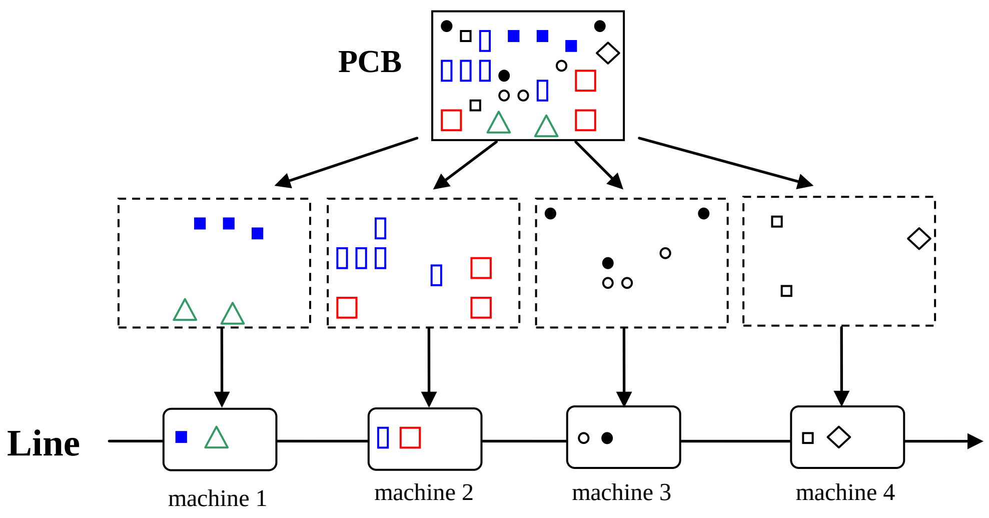


Figure 3.1 An example for the component allocation problem

A complete mathematical formulation is established in this chapter. Let $j = 1, \ldots, J$ denote the component types required by the PCB and let C_k be the feeder capacity on machine k. The following 0-1 decision variables are introduced:

x_{jk} : = 1, if the feeder for component j is set up on machine k; and = 0, otherwise.

Let x denote the component allocation represented by all x_{jk}, and $t_k(x)$ denote the process time for machine k ($k = 1, \ldots, K$) induced by the component allocation x. With X denoting the set of feasible component allocations, a mathematical model for the component allocation problem can be written as:

$$\min_{x \in X} \max_{k=1,\ldots K} t_k(x) \tag{3.1}$$

Subject to: $$\sum_{k=1}^{K} x_{jk} = 1 \quad \text{for all } j, \tag{3.2}$$

$$\sum_{j=1}^{J} x_{jk} \le C_k \quad \text{for all } k, \tag{3.3}$$

$$x_{jk} \in \{0, 1\} \quad \text{for all } j, k. \tag{3.4}$$

(Model 3-1)

Not considering the minimax operator in the objective function (3.1), the model could still be highly nonlinear due to the nonlinear relationship between $t_k(x)$, the process time for machine *k* and the component allocation *x*. In fact, the actual value of the process time for each machine is determined by the solutions to the machine optimization problems, i.e., the feeder arrangement problem and the placement sequencing problem for the machine, both of which are *NP-hard*.

Integration of the component allocation problem with these machine optimization problems will result in a very complicated model, which is difficult even for heuristic methods [Cra02]. Therefore, an effective and efficient solution strategy is required.

3.3 A SOLUTION STRATEGY

In this section, a decomposed solution strategy is proposed for solving the component allocation problem, without tackling the lower-level machine optimization problems, i.e., the feeder arrangement problem and the placement sequencing problem.

The proposed solution strategy is inspired by a similar solution strategy for solving the so-called Location Routing Problem (LRP), which is also a hierarchical planning problem in logistics [Lap88].

The location routing problem can be defined as follows: Customers distributed in a planning area are planned to be served by several facilities. A feasible set of potential facility sites and expected demands of each customer are given. Each customer is assigned to a facility which will supply its demand. The shipments of customer demand are carried out by vehicles which are dispatched from the facilities, and operated on routes that include multiple customers. There is a fixed cost associated with opening a facility at each potential site, and a distribution cost associated with any routing of vehicles, which includes the cost of acquiring the vehicles used in the routing, and the cost of delivery operations. The cost of delivery operations is linear in the total distance traveled by the vehicles. In the LRP, the objective is to simultaneously seek the optimal location of facilities, the optimal allocation of customers to facilities, and the associated minimum-cost routes to serve the customers.

Chien [Chi93] proposed a nested heuristic approach for the LRP using some TSP (Traveling Salesman Problem) optimal tour estimators, which can accurately estimate the traveling distance of the optimal tour for a TSP without solving the TSP. With an accurate and fast TSP estimator, the LRP can then be decomposed into a modified location problem with the routing costs approximated by the estimator, and a multi-depot vehicle routing problem once the location sites have been determined. It was shown that this nested approach reduces the complexity of solving the location and the routing problems simultaneously, and hence, may provide good feasible solutions to the LRP in less computation time.

The TSP optimal tour estimators have been well investigated. Most of these TSP optimal tour estimators are based on linear regression models that take into account the most important factors, including the number of visited locations and the area of the service region.

The following equation shows an example of an effective TSP tour estimator [Kwo95].

$$T^* = 2.0212d + 0.5696n\sqrt{a/n} \tag{3.5}$$

where

T^* is the estimated length of the optimal tour for a TSP,

d is the average straight-line distance from the customers to the depot,

n is the number of points (customers plus depot) in a TSP

a is area of the smallest rectangle that covers all the customer locations

It is well known that for many placement machines, the placement sequencing problem can be formulated as a TSP while the feeder arrangement of the machine is fixed [Lei89; Moy97]. It may be possible to develop a placement time estimator, which can estimate the placement (process) time without solving the machine optimization problems. Similar to the location routing problem, the component allocation problem can then be solved as a general minimax problem (Model 3-1), with the process time for each machine approximated by the placement time estimator.

Based on the above discussion, the solution strategy for the component allocation problem can be proposed as follows:

- Develop a regression-based placement time estimator which can yield accurate estimates of placement time without solving the machine optimization problems.
- Develop a specific genetic algorithm for solving the component allocation problem, with the solutions evaluated using the placement time estimator.

In the following sections, the proposed solution strategy is implemented and described in more details. First, the effectiveness of the estimator is examined by developing a placement time estimator for a turret-type placement machine. After that, a specific genetic algorithm for the component allocation problem is devised, using the placement time estimator to evaluate solutions. The effectiveness and efficiency of the solution method is examined through solving problem instances.

3.4 A REGRESSION-BASED PLACEMENT TIME ESTIMATOR

As discussed in Section 1.1, there are different types of placement machines available in the industry. The technological characteristics and operation modes may differ from a machine type to another. Therefore, specification and calibration of the placement time estimators should be different for different machine types. In this section, a placement time estimator will be developed specifically for a turret-type placement machine, Fuji CP732. Nevertheless, the methodology is general for developing the placement time estimators for different machine types.

Like the TSP tour estimator shown in (3.5), the proposed placement time estimator is based on linear regression method. For most regression applications, it requires decisions on which variables to be included in the model, the form the variables should take (for example, x, x^2, $1/x$, etc), and the functional form of the model [Raw98]. This process is called model specification. In order to specify a most suitable model, the characteristics of the investigated placement machine should be examined.

The development of the placement time estimator involves the following process. First, the placement process of the investigated machine is analyzed and the influential factors to the placement time are identified. Then, the functional form of the regression model is decided. After that, experimental tests are conducted for collecting the required data for model calibration. Finally, the model is fit on the collected experimental data, and a statistical analysis is conducted.

3.4.1 The placement process and influential factors

The Fuji CP732 placement machine is illustrated in Figure 3.2. The placement device is a turret with 16 stations. There is a placement head on each station. Each pick-and-place cycle consists of two stages. In the first stage, the placement head on station 1 picks a component from a component feeder on the feeder carrier while the placement head on station 9 places another one onto the PCB. The time for the placement head to pick a component from the positioned feeder carrier and the time for the placement head to place a component on the positioned PCB are fixed and equivalent. This time is referred to as the fixed pick and place time (*FPP*). In the second stage, the turret rotates by 22.5º. At the same time, the feeder carrier moves along X axis to locate the required feeder under station 1, and the PCB holder moves to locate the next placement location under station 9. So, the time for the second stage is decided by the longest one among the PCB movement time, feeder carrier movement time, and turret rotation time (which is usually the shortest). The aforementioned cycle is repeated until the placement for the current PCB is finished.

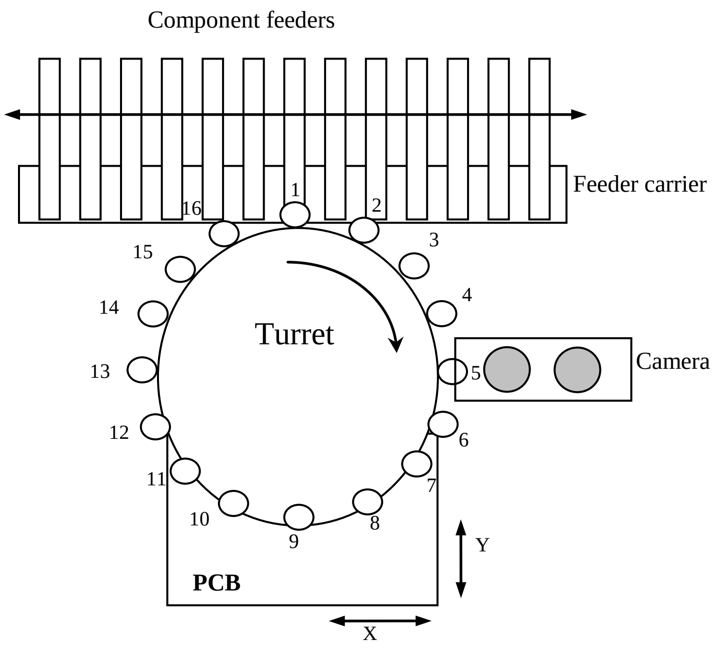


Figure 3.2 Illustration of a Fuji CP732 machine

As discussed in Chapter 1, the feeder arrangement problem (FAP), which determines the location of different component feeders on the feeder carrier, and the placement sequencing problem (PSP), which determines the component placement sequence, are two fundamental problems for machine optimization. The two problems are intertwined with each other to determine the feeder movement time and the PCB movement time in each pick-and-place cycle. The objective of the two machine optimization problem is to minimize the total process time for a PCB.

Obviously, the two machine optimization problems are highly dependent on the characteristics of the currently processed PCB, e.g., the locations and number of the components on the board. These factors are influential to the final process time, which is the objective value for the machine optimization problems. Therefore, the placement time for the PCB may be estimated through considering these factors.

Based on the above observation, the factors affecting the placement time should be included in the estimator. Firstly, the number of the pick-and-place cycles is determined by the total number of components. For this reason, the number of components to be placed may have impact on the placement time. Secondly, the closeness of the component locations on the board influences the moving time of the PCB holder and thus affects the placement time. Thirdly, the number of component types influences the number of required feeders and thus affects the moving time of the feeder carrier. These three factors are most important and should be considered in the development of the placement time estimator. Since the impacts of these factors on the placement time are different, the model should be appropriately specified to reflect these impacts.

3.4.2 Model specification

Suppose on a PCB, there are a set of N components belonging to F component types to be placed by a machine. Let A denote the area of the smallest rectangle that covers all the components. Thus, A reflects the closeness of the component locations to some extent. As discussed earlier, all the three factors, N, F, A, may influence the placement time.

However, the variable, A, may not have linear effect on the placement time. Similar to the TSP tour estimator (3.5), other two variables are proposed: $\sqrt{NA}$ and $\sqrt{NAF}$. The introduction of these two variables can be explained as follows. The term $\sqrt{A/N}$ reflects the average closeness of all the components, while the term $\sqrt{AF/N}$ reflects the average closeness of the components of the same type. In addition, it can be inferred that the marginal effect of the component closeness on the placement time may increase with the number of the components. For this reason, two interaction terms $N\sqrt{A/N}$ and $N\sqrt{AF/N}$ (that is, $\sqrt{NA}$ and $\sqrt{NAF}$) may be more appropriate as potential estimator variables than A for the linear regression model. This inference has been illustrated through preliminary results.

Totally four variables, N, F, $\sqrt{NA}$, and $\sqrt{NAF}$ are considered as candidate variables in the linear regression model. The model for the estimator with all the candidate variables is given as follows:

$$CT = b_0 + b_1 N + b_2 F + b_3\sqrt{NA} + b_4\sqrt{\mathrm{N}AF} \tag{3.6}$$

In the above model, there are 4 potential estimator variables, including different forms of the same basic variables. However, too many variables may cause overfitting, which reduces or destroys the ability of the model to generalize beyond the fitting data. In order to avoid overfitting of the regression model, the *all-possible-regressions* procedure is used, i.e., all possible regression models with every possible subsets of variables are tested and compared. The best subset of variables can then be identified. This will be discussed in Subsection 3.4.4.

3.4.3 Data collection

To calibrate the regression model, a data set is required. In this section, experimental tests are conducted to obtain the data. For this purpose, some PCBs are generated randomly and the placement time values for these PCBs are collected.

Nowadays, most of the machine vendors provide machine optimization software to determine the feeder arrangement and placement sequence for the placement machine. After deciding the feeder arrangement and placement sequence, the software can then simulate the placement process and calculate the placement time. Although the simulation process is much time-consuming, the simulated placement times are much accurate. For practical considerations, these simulated placement times obtained by the software will be used for fitting the estimating model.

It should be noted that the placement time estimator based on this data is not to estimate the optimal placement time because the machine optimization software cannot guarantee optimal solutions to the machine optimization problems. Instead, the estimator is to estimate the placement time that is achieved by the solutions to the machine optimization problems obtained by the vendor software. This is nontrivial because the machines will operate according to the feeder arrangement and the placement sequence obtained by the vendor software, rather than optimal optimization solutions, and the placement times obtained by the vendor software represent the realistic placement times in the shop floor. On the other hand, if the vender software is adjusted or new optimization methods are used, the estimator should be built again with new placement time data.

Based on the above discussion, the generated PCBs will be input into the machine software, i.e., Fuji Flexa, and relative placement time values are obtained. Before the discussion on the experiments for collecting the placement time data, the following assumptions are made.

- The speed setting for the PCB holder is at high-speed for all the components. Usually, the moving speed of the PCB holder is set to be slower for larger components in order to prevent them from slipping away from the board. However, a turret-type placement machine usually processes small components and uses a high-speed setting.
- The placement times do not include the board loading time and the fiducial time (for coordinate calibration before placement), both of which are very small and can be easily added to the final process time.

In the experiments, 100 virtual PCBs are generated. For each of the 100 PCBs, The *width* and the *length* of a PCB are generated independently and uniformly within the range [100mm, 500mm]. The number of components is generated independently and uniformly within the range [50, 800]. For each component, its *x* coordinate on the board is generated randomly within the range of [0, *width*] and its *y* coordinate is generated randomly within the range of [0, *length*].

The number of component types for a PCB is generated independently and uniformly within the range [5, 30]. In order to reflect the characteristics of realistic

PCBs, the usage pattern of different component types is not uniform. For each component type, a so-called usage frequency index is generated randomly within (0, 1). The roulette wheel selection method is used for the determination of the type for each component. Let p_i be the usage frequency index for component type i. For each component on the board, a random number R is generated within (0, $\sum p_{\mathrm{i}}$). The type of this component, t, is determined, such that:

$$\sum_{i=1}^{t\text{-}1} p_i < R < \sum_{i=1}^{t} p_i \ . \tag{3.7}$$

It should be noted that the actual number of component types used by a PCB may be less than the number of all component types, because some component types may not be used at all.

The data of the 100 PCBs are input into the vendor software, Fuji Flexa, and the placement time values are obtained. The obtained data are summarized in Table 3.1. The detailed information of the first PCB (with 61 components) is shown in Appendix I.

Table 3.1 Characteristics and placement time values (in seconds) for the 100 PCBs

PCB	*N*[a]	*F*[b]	*A*[c]	CT[d]
1	61	7	155400	12.15
2	63	14	114285	13.03
3	64	18	23210	11.5
4	66	23	48792	12.36
5	70	10	55419	11.56
6	72	16	187056	16.03
7	76	21	50304	13.43
8	82	19	88550	15.36
9	103	14	66005	16.99
10	107	6	65988	14.97
11	110	7	152852	19.31
12	112	11	51156	15.83
13	114	19	27537	16
14	120	14	206298	22.75
15	139	16	119798	24.42
16	148	6	124608	20.83
17	193	27	77714	31.78
18	213	15	68705	29.73
19	213	27	84413	34
20	216	23	123185	35.01
21	217	12	77592	28.8
22	224	13	56810	28.09
23	250	16	29280	28.79
24	255	11	65411	31.75
25	258	19	101040	38.32
26	259	10	86229	33.01
27	291	16	138408	42.34
28	291	17	66980	36.38
29	292	13	91368	37.9
30	292	9	83616	33.84
31	301	19	88105	41.35
32	311	22	177184	50.3
33	316	15	164592	47.01
34	317	26	20705	35.21
35	325	27	43803	40.83
36	325	17	39610	36.47
37	333	27	51106	43.35
38	337	28	88695	48.47
39	339	9	99880	39.35
40	340	15	139840	46.52
41	345	25	49698	44.85
42	366	15	72128	43.78
43	367	6	51339	36.12
44	372	14	37604	39.3
45	378	23	69136	49.39
46	382	15	97410	48.61
47	383	7	57620	38.25
48	386	23	177508	61.31
49	386	26	100989	54.68
50	387	21	159201	56.52
51	394	17	135954	53.95
52	398	11	72072	44.37
53	400	22	159676	60.18
54	423	24	57360	51.29
55	448	7	28890	40.04
56	463	24	45384	53.28
57	466	14	90584	54.8
58	472	23	80301	59.98
59	473	27	186333	71.24
60	503	13	88550	56.16
61	508	15	155628	66.24
62	517	14	79866	56.11
63	520	21	107797	66.1
64	533	23	176512	75.96
65	539	6	179935	57.49
66	543	16	29150	52.35
67	575	15	205480	75.06
68	582	28	56000	67.42
69	606	11	63360	61.78
70	609	8	61712	59.06
71	610	19	50851	64.14
72	610	27	144189	84.6
73	620	7	67398	58.15
74	628	14	153080	75.89
75	638	23	76544	73.9
76	643	13	184868	79.62
77	651	22	92336	76.05
78	656	23	80832	76.79
79	656	6	41334	57.88
80	656	20	163815	85.72
81	675	28	89579	82.89
82	690	14	136452	78.4
83	692	25	197918	97.92
84	710	17	35035	67.14
85	719	22	95238	83.68
86	723	6	162640	74.74
87	727	9	74868	71.23
88	730	8	62622	67.19
89	759	10	66742	72.62
90	767	7	94363	72.57
91	774	8	50440	69.84
92	777	20	51562	78.72
93	782	28	112812	96.29
94	782	21	133901	94.72
95	784	24	17760	71.41
96	784	14	46364	75.92
97	790	22	134568	95.74
98	791	10	210834	89.37
99	798	25	123060	96.86
100	800	14	67518	80.22

[a] number of components on the PCB

[b] number of component types used by the PCB

[c] area of the smallest rectangle that covers all the components

[d] placement time required by the machine to place the components

3.4.4 Model fitting and statistical analysis

The *all-possible-regressions* procedure is used for identifying the most suitable model with the best subset of the candidate variables which are discussed in Subsection 3.4.2.

All possible regression models with every possible subsets of variables are fit on the collected data and the results are compared. Table 3.2 summarizes the results for the best models with the highest R^2 (coefficient of determination) for all subset sizes, which are obtained by MINITAB 14.

Table 3.2 lists for each model the R^2, Mallows' C_p (an assessment statistic which will be discussed later), the S value (standard error of estimate), and a listing of the variables in the model. For example, the best one-variable model is a function of N and produces an R^2 value of 89.8; the best two-variable model uses N and $\sqrt{NAF}$ and has an R^2 value of 99.9; and so forth. Looking at the R^2 values, it can be seen that this statistic increases rather rapidly going from one to two variables, and changes very little as more variables are added.

To decide the most suitable model, the Mallows' C_p (Mallow, 1973) values are considered here. The C_p statistic is defined as follows:

$$C_p = \frac{SSE(p)}{MSE} - (n - 2p) + 2 \tag{3.8}$$

where

MSE is the mean squared error for the full model (3.6),

SSE(p) is the sum of squared errors for the subset model containing p predictor variables

n is the sample size.

For any given number of selected variables, larger C_p values indicate models with larger mean squared error. For any subset model with $C_p > (p + 1)$, there is evidence of bias due to an incompletely fitting model. On the other hand, if $C_p < (p + 1)$, a model is said to be overfit. It is suggested that the smallest subset model such that $2(p +1) > C_p > (p + 1)$ for all model parameter estimates works best (Hocking, 1976). Following this principle, the model with the two variables N and $\sqrt{NAF}$ is the smallest model with suitable $C_p = 3.3$ and should be the most appropriate model.

Table 3.2 Statistical results for all subset models

p (number of variables)	R^2	C_p	S	Variables			
				N	F	$\sqrt{NA}$	$\sqrt{NAF}$
1	0.898	6653.8	7.5432	√			
1	0.757	16007.4	11.651			√	
1	0.719	18516.7	12.526				√
1	0.641	61950.6	22.870		√		
2	0.999	3.3	0.9104	√			√
2	0.961	2506.9	4.7065	√		√	
2	0.937	4081.6	5.9634	√	√		
2	0.787	14015.2	10.962		√	√	
2	0.785	14188.6	11.029			√	√
3	0.999	3.1	0.90459	√		√	√
3	0.999	3.2	0.90523	√	√		√
3	0.993	373.6	2.0016	√	√	√	
3	0.787	14000.6	11.012		√	√	√
4	0.999	5.0	0.90891	√	√	√	√

The statistical results for the selected model with variables N and $\sqrt{NAF}$ are shown in Table 3.3. Based on the results, the final form of the regression model is:

$$CT = 0.533 + 0.0706N + 0.000797\sqrt{NAF} \qquad (3.9)$$

The significance value of the F statistic is less than 0.05, which means that the variation explained by the model is not due to chance. The high value of R^2 of 0.999 indicates that the regression model fits the data very well. The standard error of estimate is only 0.9104, indicating that the model has high accuracy of estimation and is suitable for estimating the placement time.

Table 3.3 Regression analysis results for the selected model by MINITAB

Predictor	Coef	SE Coef	T	P
Constant	0.5326	0.2170	2.45	0.016
N	0.0706135	0.0005198	135.85	0.000
$N\sqrt{AF/N}$	0.00079736	0.00000979	81.43	0.000

S = 0.910399 R-Sq = 99.9% R-Sq(adj) = 99.9%

Source	DF	SS	MS	F	P
Regression	2	54691	27346	32993.16	0.000
Residual Error	97	80	1		
Total	99	54772			

Figure 3.3 shows the histogram of standardized residual. The shape of the histogram approximately follows the shape of the normal distribution, showing that the normality assumption is not violated.

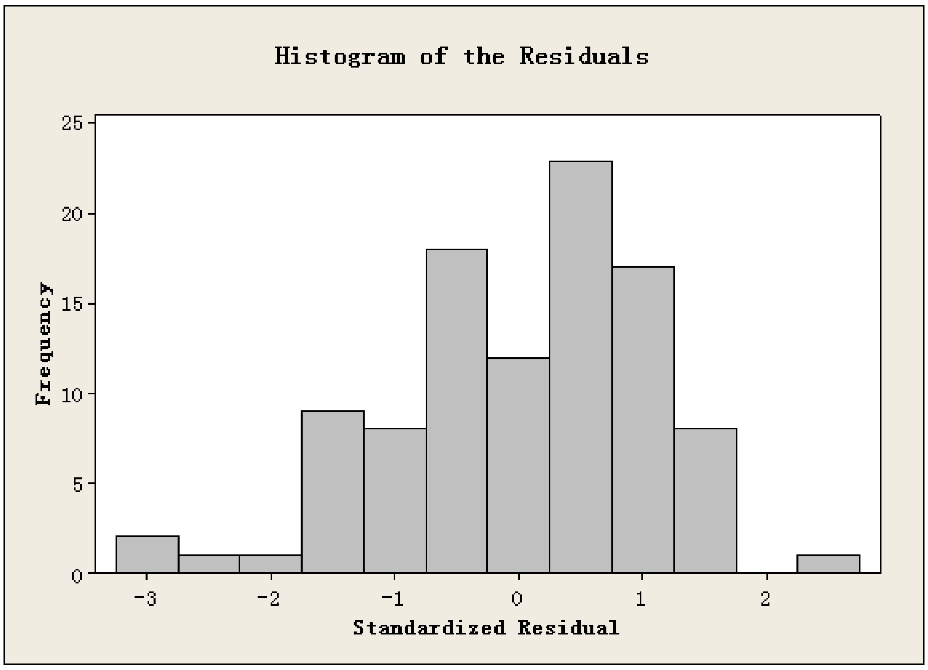


Figure 3.3 Histogram of standardized residual by MINITAB

In the literature, most researchers estimate the placement time only based on the number of components. From Table 3.2, it can be seen that the R^2 value for the model with one variable N is only 0.898, and the standard error of estimate is 7.5432, much larger than the other models with F and A, too. This indicates that considering only the component number may yield inaccurate estimate of the placement time.

From the proposed placement time estimator (3.9), some implications can also be made.

First, the placement time for processing the same number of components may differ a lot with different number of component types and different closeness degrees of component locations. For example, the placement time with N = 200, F = 10, A = 10000 mm^2 is estimated to be 17.74 seconds and the placement time with N = 200, F

= 20, A = 20000 mm^2 is estimated to be 21.30 seconds. The percentage difference between the two placement time values for the same number of components is 20%.

Second, for solving the component allocation problem, evenly balanced solution does not necessarily mean that the line cycle time is minimized. The A values, which influence the placement time estimation, are dependent on how the components are allocated to the machines. The placement time estimator developed in this research considers all the factors that affect the placement time and can yield fast and accurate estimates of placement time, without solving the machine optimization problems. It can be used to evaluate the solutions for the component allocation problem, without adding significant computational efforts.

3.5 A GENETIC ALGORITHM FOR THE CAP

Based on the placement time estimator developed in the previous section, an algorithm could be developed to solve the Component Allocation Problem (CAP) without tackling the machine optimization problems.

However, even with the process time for each machine estimated by the efficient placement time estimator, the component allocation problem, which is formulated as a minimax assignment problem (Model 3.1), is still *NP-hard* [Jip01].

In order to solve the problem efficiently, a genetic algorithm, which uses the placement time estimator for solution evaluation, is proposed.

The general process for genetic algorithms has been discussed in Chapter 2 (see Figure 2.4). The key issues in developing a genetic algorithm are the representation scheme, genetic operators, fitness evaluation, and reproduction method. In this section, these issues are described in detail, followed by the experimental tests to examine the effectiveness and efficiency of the solution methods.

3.5.1 Representation scheme

The representation scheme refers to how the solution is represented by a chromosome in the genetic algorithm. In most cases, problems can be represented in more than one way, some of which may be more amenable to evolutionary techniques than others [Dej06].

There are two primary approaches one might take in choosing a representation: a phenotypic approach in which individuals represent solutions internally exactly as they are represented externally and a genotypic approach in which individuals internally represent solutions encoded in a universal representation language. Although a genotypic approach encourages rapid prototyping of new applications, it is difficult to take advantage of domain knowledge. Therefore, a phenotypic approach, which allows for more exploitation of problem-specific properties, is used for the investigated component allocation problem.

The representation scheme of the genetic algorithm is illustrated in Figure 3.4. Each gene in the chromosome represents a feeder for a specific component type, with its value representing the machine to which the feeder is assigned.

Consider that the first PCB with 61 components and 7 component types in Table 3.1 (Detailed data for the PCB is shown in Appendix I) is processed by an assembly line with 4 placement machines. The chromosome in Figure 3.4 represents a solution to the component allocation problem in this case. For example, the third gene which represents the feeder for component type 3 is allocated to machine 2. That is, all the components of type 3 are placed by machine 2.

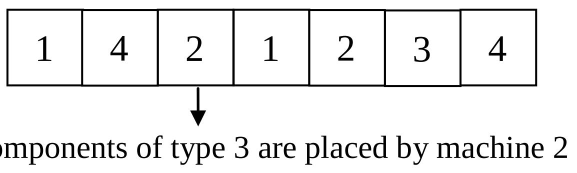


Figure 3.4 A chromosome in the proposed GA for the CAP

According to the solution represented by the chromosome in Figure 3.4, the component allocation solution for the PCB is shown in Table 3.4.

3.5.2 Genetic operators

Genetic (reproductive) operators refer to the mechanisms for generating new chromosomes from existing ones. Genetic operators are much important for the performance of genetic algorithms and should be carefully designed according to the specific characteristics of the problem. Owning to the phenotype representation scheme, the genetic operators can be more meaningful to take advantage of domain knowledge.

Table 3.4 Component allocation solution represented by a GA chromosome

Component Number	Type	X	Y	Allocated Machine	Component Number	Type	X	Y	Allocated Machine
1	1	303	167	1	32	3	102	319	2
2	5	348	310	2	33	3	278	198	2
3	4	151	297	1	34	3	352	293	2
4	5	81	327	2	35	3	395	253	2
5	6	106	109	3	36	5	344	156	2
6	2	91	347	4	37	4	340	298	1
7	2	92	90	4	38	5	22	124	2
8	1	300	271	1	39	2	210	83	4
9	5	445	118	2	40	2	156	43	4
10	1	121	196	1	41	3	286	208	2
11	2	277	341	4	42	2	394	73	4
12	5	179	346	2	43	2	247	353	4
13	2	443	121	4	44	3	121	3	2
14	1	381	90	1	45	4	171	325	1
15	3	163	45	2	46	3	363	331	2
16	1	29	211	1	47	2	239	132	4
17	5	164	342	2	48	5	91	212	2
18	4	225	39	1	49	5	180	44	2
19	2	113	294	4	50	2	90	308	4
20	7	293	244	4	51	3	61	280	2
21	2	73	13	4	52	4	243	353	1
22	5	245	212	2	53	4	151	352	1
23	4	95	112	1	54	3	255	259	2
24	5	385	148	2	55	1	434	179	1
25	5	1	251	2	56	1	397	341	1
26	1	242	233	1	57	4	274	225	1
27	1	299	18	1	58	2	1	259	4
28	3	167	209	2	59	2	105	272	4
29	4	269	148	1	60	4	81	300	1
30	2	381	117	4	61	1	269	74	1
31	2	198	260	4					

There are several common crossover operators for the genetic algorithm, including one-point crossover, multipoint crossover, and uniform crossover. Through preliminary tests, there is no significant difference among the performances of these crossover operators. Since the uniform crossover is the most common one among them, the uniform crossover is adopted in the proposed GA.

For the uniform crossover, an offspring is generated from two parents, with each gene being selected randomly from the corresponding genes of the parents. The uniform crossover can be illustrated in Figure 3.5.

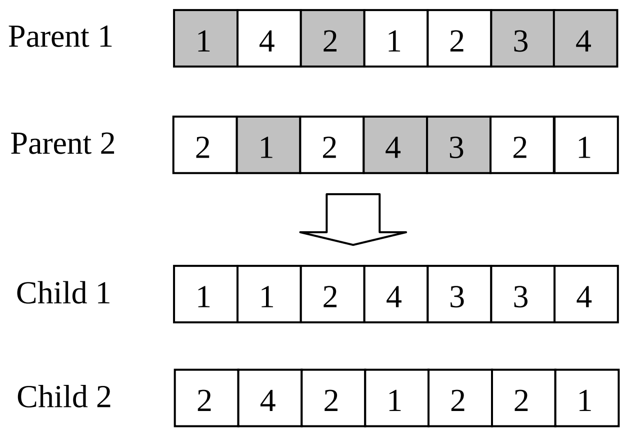


Figure 3.5 Uniform crossover

Two mutation operators will be tried and compared, i.e., the random-point mutation and the swap mutation. These two mutation operators are shown to have different performance in the proposed GA. Relative experiment results will be discussed later.

For the random-point mutation, each gene in the chromosome is changed to a random feasible value with certain possibility. The random-point mutation can be illustrated in Figure 3.6.

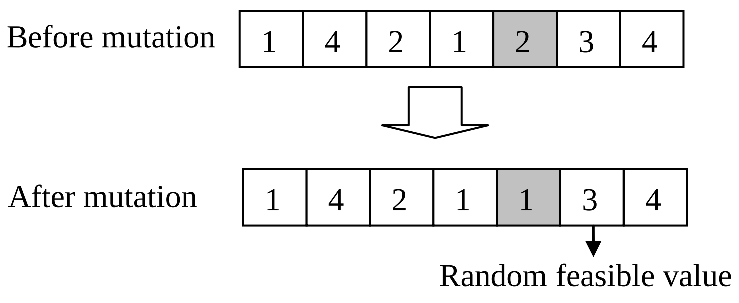


Figure 3.6 Random-point mutation

For the swap mutation, each chromosome, with certain possibility, will be adjusted through exchange of two randomly-selected genes. The swap mutation can be illustrated in Figure 3.7.

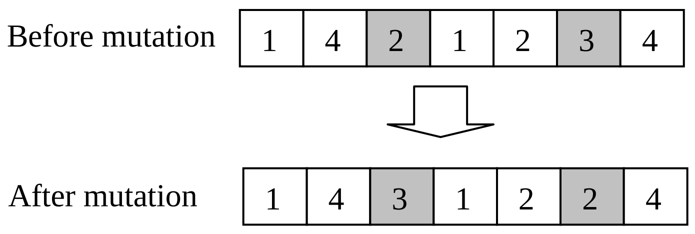


Figure 3.7 Swap mutation

Owing to the phenotype representation scheme, these two mutation operators have different practical effects on the solutions. The random-point mutation virtually adjusts the solution by reallocating a component type to a different machine. In the example shown in Figure 3.6, the component type 5, which is originally allocated to machine 2, is reallocated to machine 1 after random-point mutation. Comparatively, the swap mutation virtually adjusts the solution by exchanging component types among two machines. In the example shown in Figure 3.7, the component type 3 and 6 are originally allocated to machine 2 and 3, respectively. After the swap mutation, the component type 3 is allocated to machine 3 while the component 6 is allocated to machine 2.

Since the two mutation operators have different practical effects on the allocation solutions, their performance will be compared in the following experimental tests in Section 3.6.

3.5.3 Fitness evaluation based on the estimator

3.5.3.1 Fitness function

The placement time estimator established in Section 3.4 is used for evaluating the chromosomes in the genetic algorithm. Since the objective of the algorithm is to minimize the largest placement time of the machines, the fitness value of each chromosome could be set to the inverse of the largest placement time, that is,

$$fitness = \frac{1}{CT_{\max}} \tag{3.10}$$

where $CT_{\max}$ is the largest process time of the machines.

However, this fitness function does not give any feedback to the algorithm when two solutions have the same value of the largest placement time. In order to tackle this problem, a new fitness function is proposed:

$$fitness = \frac{1}{a * CT_{\max} + b * LT} \tag{3.11}$$

where $CT_{\max}$ is the largest process time of the machines, LT is the sum of the process times for all the machines in the line, a and b are two weighting coefficients.

Fitness function (3.11) considers reducing both the placement time for the bottleneck machine and the sum of the placement times of all machines. Coefficient *a* is much larger than coefficient *b* so that the genetic algorithm gives much higher priority to minimizing the placement time of the bottleneck machine, which is the objective of the problem. The performances of the two fitness functions (3.10) and (3.11) will be compared in the following experimental tests.

3.5.3.2 Estimation of placement time

The placement time for each machine is estimated by the established placement time estimator (3.9), which requires the calculation of the three values, i.e., the number of the components allocated to the machine, N, the number of component types, F, and the area of the smallest rectangle that covers all the allocated components, A.

While the number of the allocated components N and the number of component types F can be obtained easily, the calculation of the area of the smallest rectangle that covers all allocated components involves finding the smallest and the largest coordinates of the allocated components and thus requires nontrivial computational time.

Consider the same CAP example discussed in Subsection 3.5.1 and the chromosome illustrated in Figure 3.4. The components allocated to machine 1 can be obtained and listed in Table 3.5.

Table 3.5 List of components allocated to machine 1

Component Number	Type	X	Y
1	1	303	167
3	4	151	297
8	1	300	271
10	1	121	196
14	1	381	90
16	1	29	211
18	4	225	39
23	4	95	112
26	1	242	233
27	1	299	18
29	4	269	148
37	4	340	298
45	4	171	325
52	4	243	353
53	4	151	352
55	1	434	179
56	1	397	341
57	4	274	225
60	4	81	300
61	1	269	74

From the table, it can be easily seen that the number of the allocated components $N = 20$, and the number of component types $F = 2$. To obtain the value of A, the smallest and largest coordinates of the components should be determined. From the table, it is found that the smallest X and Y coordinates are 29 and 18, respectively, and the largest X and Y coordinates are 434 and 353, respectively. Therefore, $A = (434 - 29) \times (353 - 18) = 135675$.

The placement time for machine 1 can be estimated by estimator (3.9) as follows.

$CT = 0.533 + 0.0706N + 0.000797\sqrt{NAF}$ = 0.533 + 1.412 + 1.857 = 3.802 seconds.

For a realistic case, the number of components allocated to a machine may be several hundred. In some preliminary experimental tests, it is found that the solution evaluation occupies the majority of computation time of the algorithm. Since the calculation of *A* values requires nontrivial computational efforts, in order to improve the efficiency of the genetic algorithm, the following method is proposed to calculate the *A* values.

At the beginning of the genetic algorithm, some preliminary calculations are conducted, that is, for each component type *i*, the number of components belonging to type *i*, and the smallest and largest coordinates for the components of type *i* are determined. Table 3.6 summarizes the preliminary calculation results for the problem.

With these preliminary calculation results, the computational efforts required for solution evaluation in the GA can be significantly reduced. Consider the same example discussed above, knowing that component types 1 and 4 are allocated to machine 1, the values of *N* and *A* for machine 1 can be easily obtained as follows:

$N = N_1 + N_4 = 10 + 10 = 20$;

$A = (\max(X_{\max}^1, X_{\max}^4) - \min(X_{\min}^1, X_{\min}^4)) \times (\max(Y_{\max}^1, X_{\max}^4) - \min(Y_{\min}^1, Y_{\min}^4)) = (434 - 29) \times (353 - 18) = 135675.$

Table 3.6 Preliminary calculations for the GA

Component Type	Number of Components N_i	X_{min}	Y_{min}	X_{max}	Y_{max}
1	10	29	18	434	341
2	16	1	13	443	308
3	11	61	3	395	319
4	10	81	39	340	353
5	12	1	44	445	346
6	1	106	109	106	109
7	1	293	244	293	244

In the preliminary experiments, it is found that this method for calculating the parameters for the placement time estimator can save more than 90% of computational time for the proposed genetic algorithm.

3.5.4 The general framework

The selection mechanisms are the driving force for genetic algorithms. There are two places in genetic algorithms where a selection occurs: when choosing parents to produce offspring, and when choosing which individuals will survive. If the combined selection pressure is too strong, the genetic algorithm is likely to converge too quickly to a suboptimal region of the space. As a consequence, usually one of the two selection processes (i.e., either parent or survival selection) adopts random selection method, while the other adopts fitness-based selection method [Dej06].

In the proposed GA, the parents are randomly selected. The replace-worst replacement strategy is adopted for selecting the chromosomes for survival, that is, a new generation is formed by selecting the best individuals from the parents and the offspring. The general framework of the proposed genetic algorithm can be illustrated in Figure 3.8.

The general process for the proposed GA is described as follows.

At the beginning of the algorithm, an initial population of chromosomes is randomly generated. This procedure consists of randomly allocating the component types to machines in the line.

Then, for each generation, pairs of chromosomes are randomly selected and the crossover operator is applied to each pair of parents to produce offspring. The number of parents is determined by the crossover rate C_r, which represents the percentage of population to be chosen as parents. The generated children are then mutated according to a certain mutation possibility M_r. For the random-point mutation, M_r represents the possibility for each gene in a chromosome to be changed. For the swap mutation, M_r represents the possibility for a chromosome to be selected for mutation.

After the reproductive operations, the best chromosomes from the original chromosomes and the offspring are selected for survival in the next generation.

The genetic algorithm is run until there is no improvement on the best found solution during a certain number of generations.

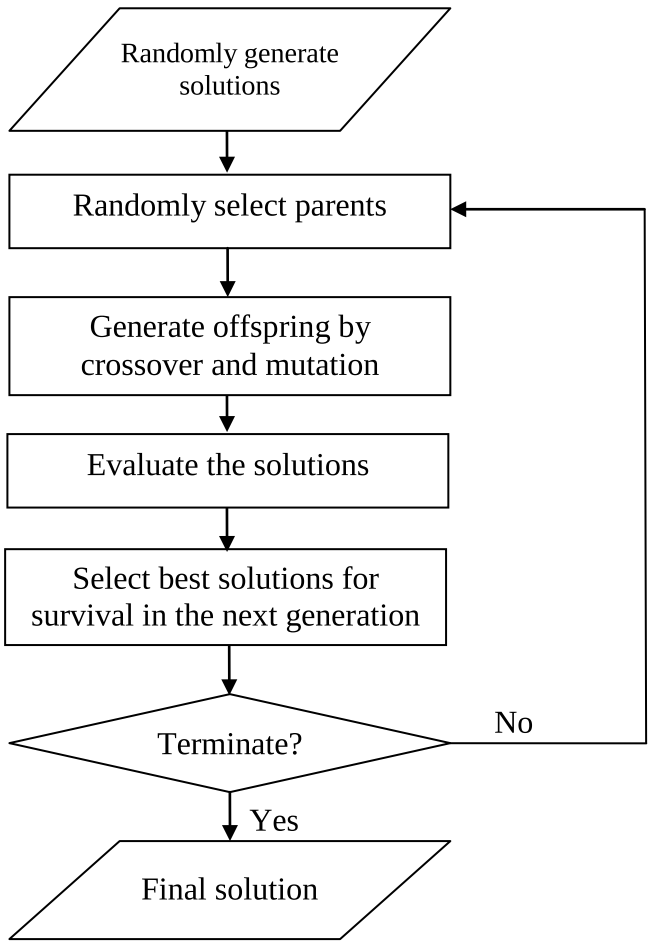

Randomly generate solutions
Randomly select parents
Generate offspring by crossover and mutation
Evaluate the solutions
Select best solutions for survival in the next generation
Terminate?
No
Yes
Final solution

Figure 3.8 General framework for the proposed GA for the CAP

3.6 EXPERIMENTAL TESTS

This section is devoted to the experimental tests of the proposed solution method for the component allocation problem. The experiments are divided into two parts: the experiments for improving performance of the genetic algorithm, and the experiments for evaluating the effectiveness and efficiency of the proposed solution method for solving the component allocation problem.

3.6.1 Experiments for improving GA performance

As discussed in Section 3.5, two different mutation operators and two different fitness functions are proposed for the genetic algorithm. Their performance in the algorithm is compared through experimental tests. The algorithms are coded in Microsoft Visual C++ 2005, and run on a desktop computer with Pentium IV 2.26 GHz CPU and 1 GB RAM.

3.6.1.1 Problem instances

In the experiments, a set of 10 realistic PCBs are generated and used for evaluating the performance of the proposed GA method for the component allocation problem. The PCBs are assumed to be produced in an assembly line which consists of four CP732 placement machines.

The PCBs are generated in a similar way as discussed in Subsection 3.4.3, with realistic sizes comparable to those in a telecommunication product manufacturer. For each PCB, the number of components is generated independently and uniformly within the range [800, 1000]. The *width* and the *length* of the PCB are generated independently and uniformly within the range [400mm, 600mm]. The number of component types is generated independently and uniformly within the range [50, 70]. The locations of components are randomly generated within the board size. The characteristics of the generated PCBs are shown in Table 3.7.

Table 3.7 Characteristics of the 10 PCBs

PCB	N[a]	F[b]	L[c]	W[d]
1	831	51	483	429
2	851	58	459	447
3	861	64	446	414
4	873	63	494	524
5	881	61	589	407
6	914	52	444	494
7	925	62	521	473
8	944	60	438	404
9	950	55	532	576
10	960	66	538	530

[a] Number of components on the PCB

[b] Number of component types used by the PCB

[c] Length of the PCB

[d] Width of the PCB

3.6.1.2 Parameter setting for the GA

The GA parameters, i.e., the population size, crossover rate, and mutation rate, should be determined first.

The parameters for the genetic operators are determined through trial and error. The crossover rate C_r is set to 0.8. That is, 80% of the population members are randomly selected as parents to produce 80 children in each generation. For the random-point mutation, every gene in a child will be randomly changed with possibility of 0.02. For the swap-mutation, each child will be selected to be mutated with the possibility of 0.005.

The population size, i.e., the number of chromosomes in each generation, can be viewed as a measure of the degree of parallelism of the searching process, in the sense that each chromosome represents an independent agent exploring a particular area of the solution space. As the fitness landscape becomes more complex with multiple peaks, discontinuities, etc., more parallelism is required [Dej06]. However, as the population size is larger than a suitable value, its contribution to the algorithm performance becomes trivial with respect to the additional computational time.

As stated by De Jong [Dej06], a GA is not highly sensitive to moderate changes in the population size. Therefore, experimental tests are conducted to choose a suitable population size from several values, i.e., 10, 50, 100, 150, and 200. The uniform crossover, the swap mutation, and the fitness function (3.11) are used in this experiment. The GA is tested on the problem for the smallest PCB with 831 components. The algorithm is run for 4000 generations.

Figure 3.9 shows the convergence process for multiple GA runs with different population sizes. It can be seen that the GA with population size of 100 achieves much better solutions than the GAs with population size 10 and 50. However, when the population size increases from 100 to 200, the algorithm converges at a similar solution value. Table 3.8 shows the average results for 10 runs. It can be seen that the computation time spent by the GA with population size of 200 is significantly larger than that spent by the GA with population size of 100, while the objective values for the two GAs are almost the same. Similar observations are also obtained for other problem instances. Therefore, 100 is considered to be a suitable population size for the proposed GA to solve the test instances.

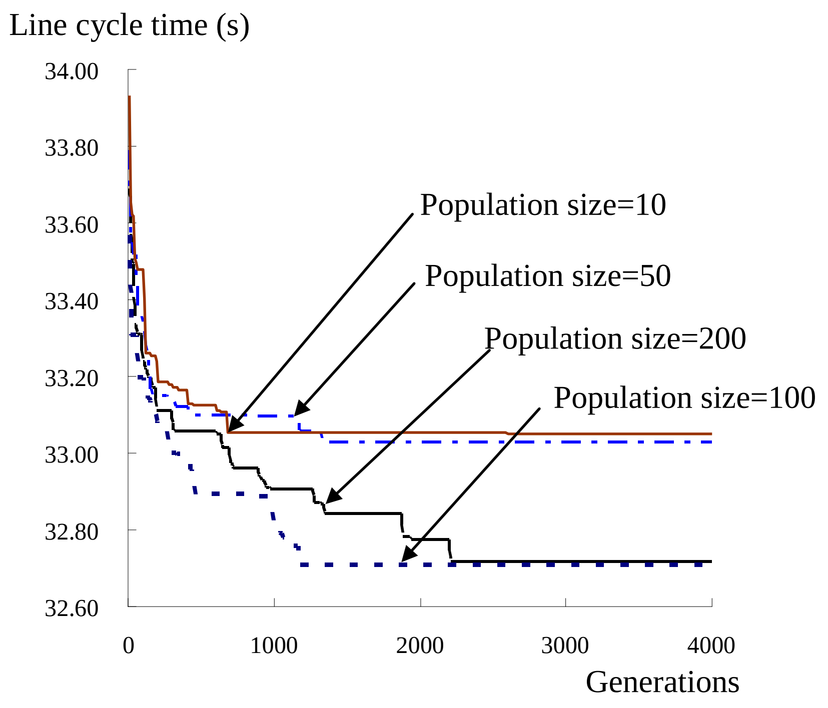


Figure 3.9 Convergence process for the GAs with different population sizes

Table 3.8 Experimental results for different population sizes

	Population size =10	Population size =50	Population size =100	Population size =200
Objective value	32.96	33.03	32.71	32.70
CPU time	4.2	9.7	17.7	52.6

3.6.1.3 Selection of genetic operators

The performance of the two mutation operators, i.e., the random-point mutation and the swap mutation, are compared in the following experiments.

In the experiments, two GAs with different mutation operators are tested. In both GAs, the fitness function (3.11) is used. Each GA is run 10 times for each of the 10 instances. For each run, the algorithm is not terminated until there is no improvement on the best found solution during 4000 generations. Figure 3.10 compares the objective values obtained by the two GAs.

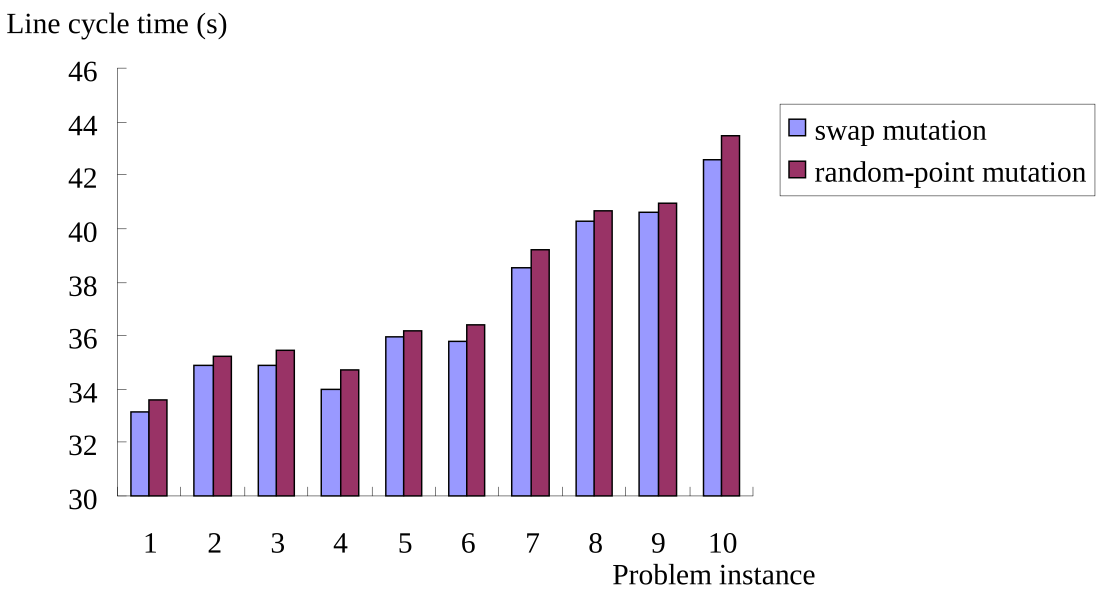


Figure 3.10 Experimental results for two mutation operators

It is found that all the results obtained by using swap mutation are better than those obtained by using random-point mutation. The average percentage reduction in line cycle time is about 1.4%.

This result is due to line balancing characteristics of the component allocation problem. The two mutation operators have different practical modification to the chromosomes. As discussed in Subsection 3.5.2, the random-point mutation adjusts the solution by reallocating a component type to a different machine, while the swap mutation adjusts the solution by exchanging component types among two machines. For an already good solution that survives into the late period of the algorithm, a reasonable way for reducing the line cycle time is to exchange components between two machines, rather than reallocate a component type from a machine to another machine. For this reason, the swap mutation may have a greater chance of reducing the line cycle time than the random-point mutation.

Because the GA with the swap mutation achieves better solutions than the GA with the random-point mutation, the swap mutation is used in the proposed GA for solving the component allocation problem.

3.6.1.4 Selection of fitness function

Here, experiments are conducted to compare the performance of the two fitness functions (3.10) and (3.11). Two genetic algorithms with these two fitness functions are tested.

For the fitness function (3.11), the weighting coefficient for the bottleneck placement time, a, and that for the total placement time, b, are set to be 1 and 0.001, respectively. Swap mutation operator is used in both algorithms. Each GA is run 10 times for each of the 10 instances. Figure 3.11 compares the average objective values obtained by the two GAs.

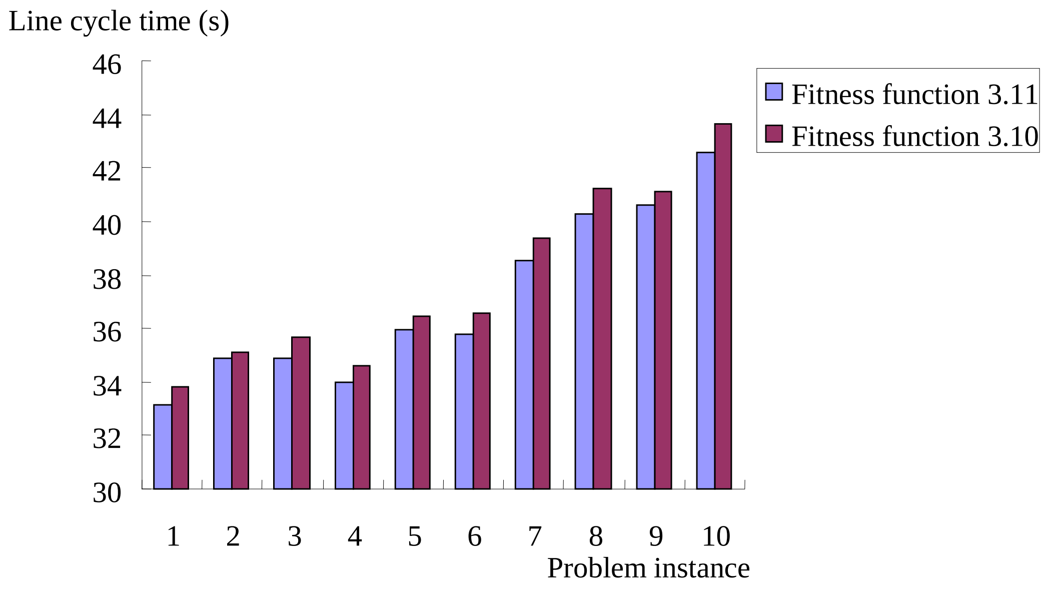


Figure 3.11 Experimental results for two fitness functions

From the results, it is found that all the objective values (line cycle times) obtained by using function (3.11) are better than those obtained by using function (3.10). The average percentage reduction in cycle time is about 1.9%.

This result is also due to line balancing characteristics of the component allocation problem. The fitness function (3.11) is helpful in providing differential feedback to the genetic algorithm even for the solutions that have the same bottleneck placement time but different total placement time. In this way, the genetic algorithm may reduce the total placement time for the machines even it cannot reduce the process time for the bottleneck machine at a certain stage.

Because the GA with fitness function (3.11) achieves significantly better solutions than the GA with fitness function (3.10), the fitness function (3.11) is used in the proposed GA for solving the problem.

3.6.2 Experiments for evaluating the solution method

In this subsection, experiments are conducted to examine the effectiveness of the proposed GA method for the component allocation problem.

3.6.2.1 Evaluation of the solutions with actual placement times

Because the objective values (line cycle time) in the genetic algorithm are based on the placement times approximated by the placement time estimator, in order to evaluate the realistic quality of the solutions, the actual placement times should be used. For this end, the component allocation solutions are input into the machine vendor software, Flexa, and the actual placement time for each machine is obtained. Then, based on the actual placement time for each machine in the line, the actual line cycle time can be obtained.

Table 3.9 summarizes the results for the 10 problem instances obtained by the proposed GA. The results are the averages of 10 runs. From the table, it can be seen that the GA solves the problems in a very short time. The average CPU time is 16.34 seconds. This shows that the proposed GA based on the placement time estimator can solve the component allocation problem efficiently.

The estimated cycle times are very close to the actual cycle times obtained by Flexa. The percentage difference is only 1.30%, indicating the placement time estimator is much effective in estimating the placement time values.

Table 3.9 Solutions to the CAP instances

PCB	*N* [a]	*CPU* [b]	*CT-GA* [c]	*CT-real* [d]	*diff* [e]
1	831	11.06	32.62	33.52	2.68%
2	851	15.02	34.41	34.89	1.38%
3	861	15.87	34.36	34.58	0.64%
4	873	19.61	33.47	33.81	1.01%
5	881	14.51	35.46	35.78	0.89%
6	914	12.29	35.28	35.84	1.56%
7	925	16.41	38.01	38.59	1.50%
8	944	18.57	39.80	40.17	0.92%
9	950	17.72	40.11	40.43	0.79%
10	960	22.36	42.09	41.42	1.62%
Average		16.34	36.56	36.90	1.30%

[a] Number of components

[b] CPU time for the genetic algorithm

[c] Estimated cycle time obtained by GA (excluding the board loading time)

[d] Real cycle time obtained by GA (excluding the board loading time)

[e] Absolute percentage difference between estimated cycle time and real cycle time

3.6.2.2 Comparison with machine vendor software

The software of the machine vendor also provides solutions to the component allocation problem (which is referred to as line balancing problem in the software). The solutions obtained by the vendor software will be used as benchmark solutions.

Before the comparison, the approach adopted by the software is discussed. In the machine vendor software, the machine optimization problems are solved using simple heuristics, so that the component allocation problem can be solved in combination with solutions to the machine optimization problems. Due the integration of the component allocation problem with the solutions to the machine optimization problems, the heuristic used by the software for solving the component allocation problem is rather simple. The heuristic can be described as the following steps: (1) Allocate one unassigned component type with the largest component number to each machine, (2) For each machine and the currently allocated components, solve the machine optimization problems and calculate the placement time for each machine, (3) Allocate one unassigned component type with the largest component number to the machine with the smallest placement time, and (4) Update the placement time of the machine in Step (3) by solving the optimization problems again, and (5) Repeat step 3 and 4 until all the component types are allocated. This simple heuristic gives priority to first allocating those component types with the largest quantities, so that the line can be well balanced at the end. However, as discussed in Section 3.4.4, a well-balanced solution does not necessarily mean that the minimal cycle time can be achieved. Other factors like the number of component types and the closeness of the component locations should also be considered.

In order to evaluate the effectiveness of the proposed solution method, the results are compared with those obtained by the vender software. The results are shown in Table 3.10. For all instances except PCB 6, the actual cycle times obtained by the GA are smaller than those obtained by the vendor software. The overall reduction in line cycle time is 0.82%.

Although the improvement achieved by the proposed method is not so great, the result is encouraging because the GA obtains the solutions without calculating exact placement times through simulation, as the vendor software does. The placement time simulation requires knowledge of technological characteristics of the machine, e.g., the moving speed of the PCB holder, the rotation speed of the turret. Due to severe competition, most machine vendors are reluctant to release these parameters.

Table 3.10 Comparison between GA solutions and vendor software solutions

PCB	*CT*-real [a]	*CT*-vendor [b]	*CT-diff* [c]	*LT*-real [d]	*LT*-vendor	*LT-diff* [f]
1	33.52	33.86	1.00%	130.28	132.22	1.47%
2	34.89	35.20	0.88%	136.74	139.01	1.63%
3	34.58	35.11	1.51%	136.83	139.15	1.67%
4	33.81	33.93	0.35%	131.31	133.69	1.78%
5	35.78	36.01	0.64%	140.72	142.61	1.33%
6	35.84	35.78	-0.17%	141.02	142.53	1.06%
7	38.59	39.01	1.08%	152.71	154.33	1.05%
8	40.17	40.54	0.91%	156.59	159.85	2.04%
9	40.43	40.87	1.08%	158.69	160.40	1.07%
10	41.42	41.84	1.00%	162.21	165.69	2.10%
Avg.	36.90	37.22	0.82%	144.71	146.95	1.52%

[a] Real cycle time obtained by GA (excluding the board loading time)

[b] Cycle time obtained by vender software (excluding the board loading time)

[c] Percentage improvement of *CT*, i.e., (*CT*- vendor − *CT*-real) / *CT*- vendor

[d] Total process time of all machines obtained by the genetic algorithm

[e] Total process time of all machines obtained by the vendor software

[f] Percentage improvement of *LT*, i.e., (*LT*-vendor − *LT*-real) / *LT*-vendor

The columns *LT-real* and *LT-vendor* in Table 3.10 show the total placement time of all machines for the GA solution and the software solution, respectively. It can be seen that the percentage reduction of the total process time are generally greater than the reduction of the line cycle time.

To explain this, consider the instance of PCB 1. Figure 3.12 shows the GA solutions for the instance with PCB 1. In the figure, the “Estimated solution” shows the placement times estimated by the placement estimator. It can be seen that the workload is well balanced over the machines with the estimated placement times. The bottleneck machine is machine 3 whose placement time is 32.62 seconds. The “Actual solution” shows the actual placement times for the GA solution. It can be seen that the difference among the actual placement times is much greater than the difference among the estimated placement times. This is caused by the estimation error of the placement time estimator. Considering the actual placement times, the bottleneck machine is machine 1, for which the placement time is 33.52 seconds. This indicates that the estimation error, though small, deteriorates the allocation solution to some extent.

If the actual placement times are known, the GA solution can be manually adjusted as the following. First, on machine 1, which is the bottleneck, find the component type that has fewest components. Then, allocate this component type to machine 2, of which the actual placement time is smallest. After the adjustment, the bottleneck machine becomes machine 4, of which the actual placement time is 33.20 seconds. This experiment indicates that the GA solutions may be easily improved by obtaining the actual placement times at the end of the GA.

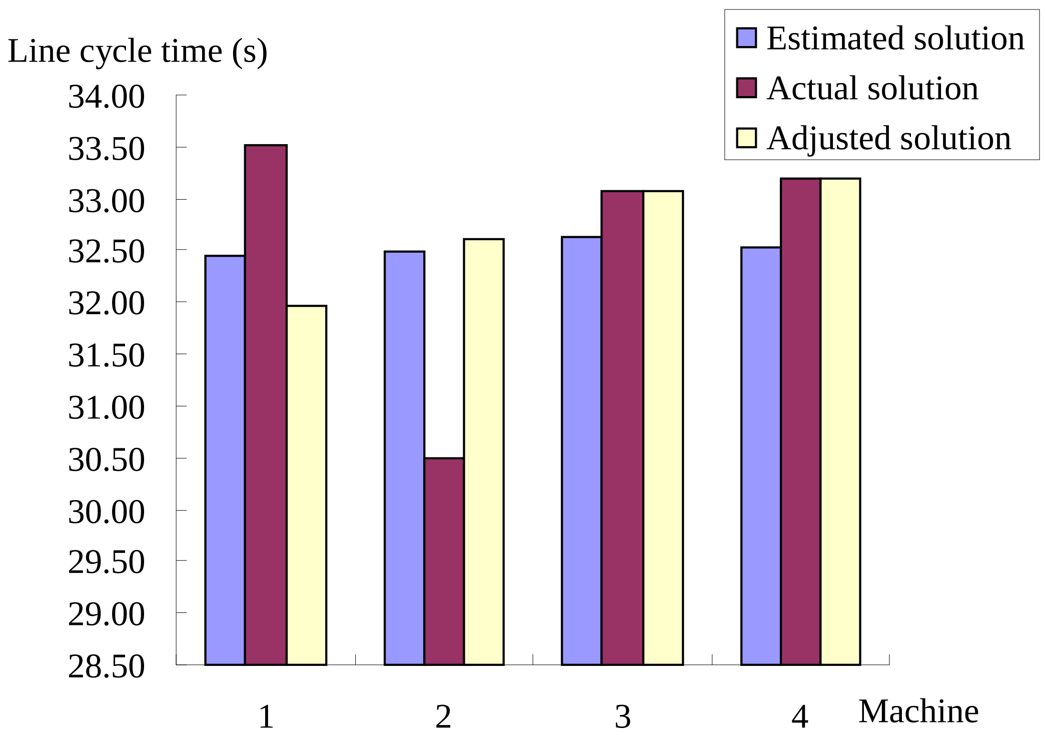


Figure 3.12 Analysis of GA solutions for the case with PCB 1

3.6.2.3 Experiments on PCBs with clustered component locations

For some realistic PCBs, the components may be clustered in location. The experiments in this subsection are conducted to examine whether such a property may affect the performance of the proposed solution method for solving the component allocation problem.

For this purpose, another set of 10 PCBs is generated. Different from the PCBs in the original set of PCBs, the components of each type are clustered in location to some extent. First, a virtual center location (x_i, y_i) is randomly generated for each component type *i*. Then, for each component belonging to type *i*, an initial location (x_0, y_0) is generated randomly. Then the location for this component is adjusted to be $(\frac{x_0 + x_i}{2}, \frac{y_0 + y_i}{2})$. In this way, the same components are clustered.

Figure 3.13 illustrates the process for generating the component locations. In the figure, components 1, 2, and 3 are belonging to type *i*. The initial locations for these components are generated randomly, as shown in Figure 3.13 (a). Then, a virtual center is generated randomly for the components of type *i*. The locations for the three components are adjusted to new locations which are closer to the virtual center point. Figure 3.13 (b) shows the new locations for the components after the adjustment.

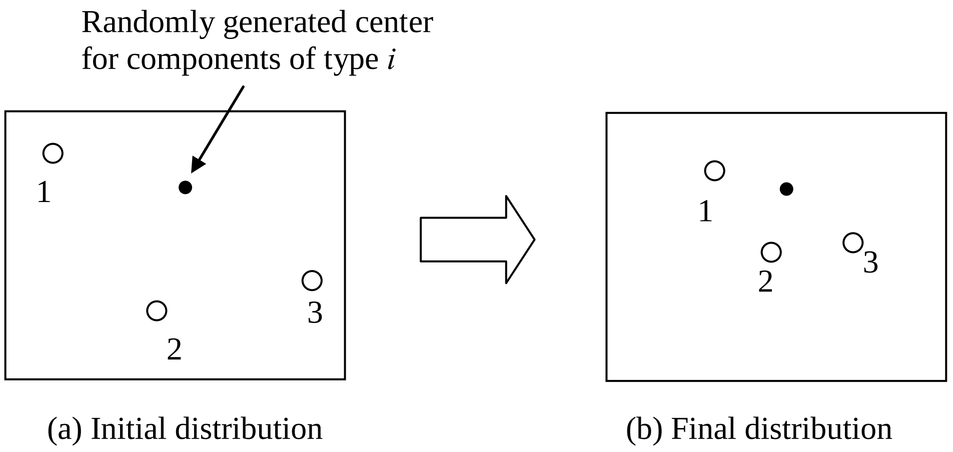


Figure 3.13 Process for generating the clustered component locations

Both the proposed GA and the machine vendor software are used for solving the new problem instances. The results are summarized in Table 3.11.

For all instances, the real cycle times obtained by the GA are significantly smaller than those obtained by the vendor software. The overall percentage improvement is 2.48%, which is much larger than that for the original instances. The greater improvement for the new instances is due to the non-uniform distribution of the components, which can be exploited by the GA to allocate more closely located components to the same machines. This can be illustrated by Figure 3.14, which shows the allocation results for PCB 11. The GA solution allocates closely located components to each machine, while the vendor software obtains the solution without considering the component locations.

Table 3.11 Comparison between GA solutions and software solutions for PCBs with clustered locations

PCB	*CT*-real [a]	*CT*-vendor [b]	*CT-diff* [c]	*LT*-real [d]	*LT*-vendor [e]	*LT-diff* [f]
11	25.77	26.94	4.34%	101.50	106.03	4.27%
12	30.97	31.72	2.36%	119.62	123.63	3.24%
13	27.10	27.73	2.27%	105.54	108.77	2.97%
14	28.26	29.58	4.46%	112.30	115.58	2.84%
15	30.21	31.08	2.80%	116.49	121.82	4.38%
16	29.13	29.46	1.12%	112.67	115.74	2.65%
17	30.56	31.38	2.61%	118.80	122.92	3.35%
18	31.93	32.76	2.53%	126.63	128.95	1.80%
19	30.33	30.72	1.27%	116.67	120.71	3.35%
20	32.32	32.66	1.04%	124.95	128.13	2.48%
Avg.	29.66	30.40	2.48%	115.52	119.23	3.13%

[a] Real cycle time obtained by GA (excluding the board loading time)

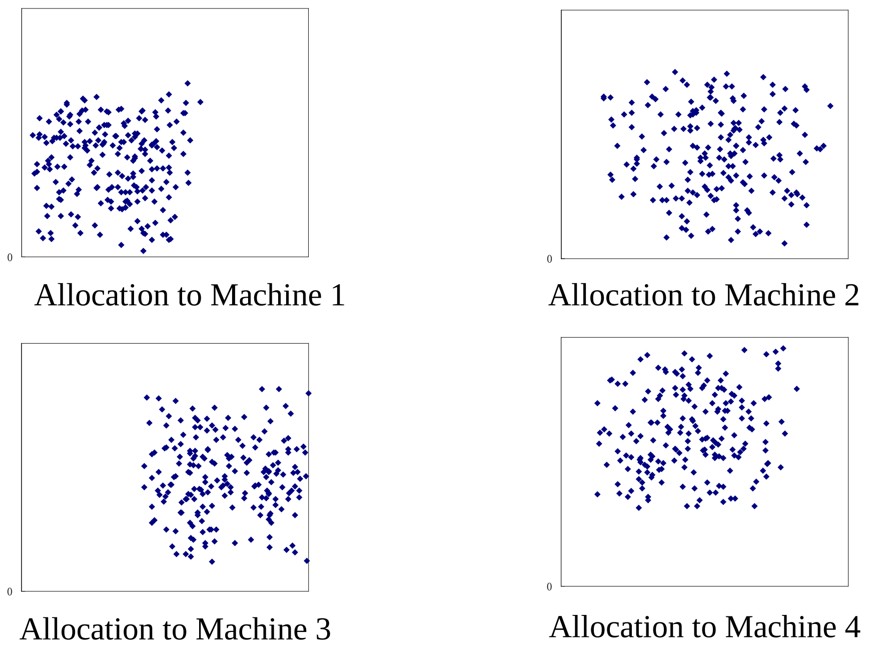
[b] Cycle time obtained by vender software (excluding the board loading time)

[c] Percentage improvement of *CT*, i.e., (*CT*-vendor − *CT*-real) / *CT*-vendor

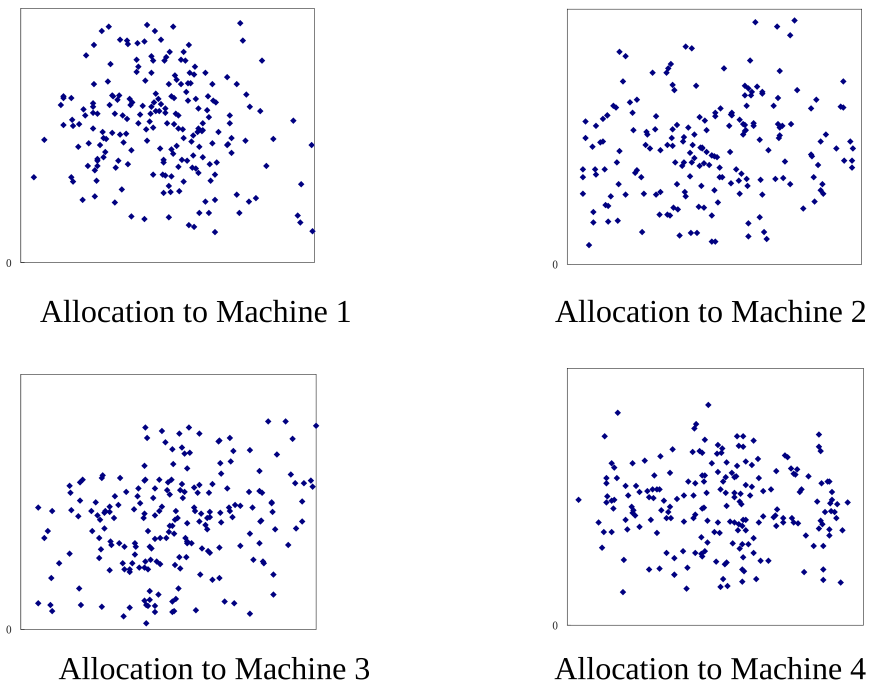
[d] Total process time of all machines for GA solution

[e] Total process time of all machines for vendor software solution

[f] Percentage improvement of *LT*, i.e., (*LT*-vendor − *LT*-real) / *LT*-vendor



(a) GA solution



(b) Vendor software solution

Figure 3.14 Allocation solutions by GA and vendor software for PCB 11

3.7 SUMMARY

This chapter has investigated the component allocation problem (CAP) in PCB assembly and proposed an effective method for the problem. Some remarks can be summarized as follows.

1. The component allocation problem in PCB assembly, which is to allocate components of a PCB to different placement machines in an assembly line, is much complicated due to its dependency on the solutions to the machine optimization problems for each machine. It is infeasible to solve the component allocation problem and the machine optimization problems simultaneously due to great computational complexity.
2. A solution strategy has been proposed to solve the component allocation problem and the machine optimization problems in a decomposed manner. The solution strategy relies on a placement time estimator that can estimate the placement time for each machine without solving the machine optimization problems.
3. A placement time estimator for a turret-type machine, i.e., Fuji CP732, has been established in this research. The placement time estimator is based on the linear regression approach. The regression model considers all the influential factors that may affect the placement time, including the number of components, number of component types, and closeness of the components. The regression model is specified based on observations of the operation mode, and calibrated using a set of experimental data. Statistical analysis shows that the placement time estimator has very high R^2 of 0.99, showing

that the estimator can yield accurate estimates of placement time. The significance value of the *F* statistic is less than 0.05, which means that the variation explained by the model is not due to chance.

4. An analysis based on the proposed placement time estimator shows that the placement times estimated only by the number of components are inaccurate and thus not suitable to be used in the algorithms for solving the component allocation problem. As most of existing approaches for the component allocation problem adopt rough estimates of placement time based only on the number of components, the relative solutions cannot be good enough. Besides the number of components, the number of component types and the closeness of the component locations should also be considered for estimating the placement times.

5. A specific genetic algorithm, which uses the established placement time estimator for solution evaluation, has been proposed to solve the component allocation problem. Using the placement time estimator, the algorithm considers all the influential factors implicitly when solving the component allocation problem. The mutation operator and fitness function are found to be important for the performance of the GA in solving the component allocation problem. The GA using the swap mutation operator achieves the average cycle time 1.4% shorter than that achieved by the GA using the random-point mutation, while the GA with fitness function considering both the line cycle time and the total line time achieves the average cycle time 1.9% shorter than that achieved by the GA with fitness function considering

only the line cycle time. These improvements are due to the line balancing characteristics of the component allocation problem.

6. Experimental results show that the proposed GA can solve the component allocation problem effectively and efficiently and achieve better solutions than those obtained by the software provided by the machine vendor. The results are encouraging, especially because the GA obtains the solutions without calculating the placement times through simulation, as the vendor software does. Even better solutions could be expected if the GA solutions are further improved by some adjustments based on the simulated placement times. The experiments also show that the component clustering characteristics of the PCB can be exploited by the proposed GA method to obtain better solutions.

The method proposed in this chapter has shown to be able to solve the component allocation problem effectively and efficiently and improve the production efficiency for a PCB assembly line. The component allocation problem presumes that a particular batch of PCB is assigned to and produced by an assembly line. In the next chapter, the scheduling problem for multiple PCB batches and multiple assembly lines, i.e., the Multi-Line Scheduling Problem (MLSP), is discussed and investigated.

CHAPTER 4

THE MULTI-LINE SCHEDULING PROBLEM (MLSP)

4.1 INTRODUCTION

The component allocation problem which arises when a batch of PCB is processed by an assembly line has been discussed and investigated in Chapter 3. On a higher planning level, a planning problem should be solved to schedule different PCB batches on multiple assembly lines (see Section 1.2.1).

For the scheduling problem in a multi-line PCB assembly shop, which is referred to in the following as the Multi-Line Scheduling Problem or MLSP, several unique characteristics need to be considered, which make the MLSP different from other scheduling problems. The characteristics for the multi-line scheduling problem in PCB assembly can be described as follows. First, the process time for each job depends on the assembly line it is assigned to. That is, the process time is line-dependent. Second, the setup time (or transition time) for a job depends on the job previously processed on the line. That is, the setup time is considered to be sequence-dependent. Third, there may be precedence requirement between the jobs. Forth, each job has its ready time and due date. Similarly, each assembly line may have its ready time. Fifth, the objective of the problem should consider both production efficiency and due date satisfaction. As discussed in Chapter 2, this specific scheduling problem has not been investigated in the literature.

In this chapter, a complete mathematical model for the Multi-Line Scheduling Problem (MLSP) in PCB assembly is established. The proposed model explicitly considers line dependent process times, ready time and due date constraints for the jobs, sequence dependent setup times, and precedence constraints between jobs. The objective of the model considers both the satisfaction of due date requirement and improvement of production efficiency. Experimental tests on solving some problem instances are conducted to verify the established model.

Due to the great complexity of the problem, a genetic algorithm is proposed. The efficiency and effectiveness of the GA are examined through both generated test instances and a realistic case study.

This chapter is organized as follows: In Section 4.2, a Mixed Integer Linear Programming (MILP) model for Multi-Line Scheduling Problem (MLSP) is developed. To verify the model, optimal solutions to some small problem instances are obtained through using an existing LP solver. Section 4.3 describes the development of the specific genetic algorithm for the MLSP, followed by the experimental tests in Section 4.4. A case study is conducted and described in Section 4.5. Finally, Section 4.6 summarizes the main work in this chapter.

4.2 A MATHEMATICAL MODEL AND EXACT SOLUTIONS

Mathematical modeling is to describe a problem in a mathematical way, and is a significant activity for better understanding and analyzing the problem. In such a mathematical way, much of the ambiguity and imprecision in verbal communication can be overcome. Meanwhile, an effective mathematical model can help to capture the essential features of the problem and provide considerable insights into the problem. Furthermore, the optimal solutions to some problem instances can be the benchmark for the heuristic solutions.

Since there is no existing mathematical model for the investigated multi-line scheduling problem, this section is devoted to the development of a mathematical model for the problem.

4.2.1 Description of the MLSP

During a planning horizon, there are a set of PCB batches to be proposed on multiple assembly lines. For most PCB manufacturers, meeting the due date requirement is the paramount objective in order to maintain the customer satisfaction. At the same time, in order to improve the production capacity with limited expensive assembly equipment, it is vital for the manufacturers to achieve high production efficiency, which is usually represented by a short makespan for the jobs. Therefore, the objective for the Multi-Line Scheduling Problem (MLSP) considers both due date satisfaction and makespan reduction at the same time.

The constraints for the MLSP may differ from a manufacturing environment to another. The investigated multi-line scheduling problem arises from a particular manufacturer of telecommunication products. Some assumptions and constraints for the problem are described as follows.

- The PCBs are produced in batches. Each batch consists of identical PCBs with the same ready time and due date. Processing of a batch is called a job. Processing of the same boards but different sides (front sides and back sides) are considered as two different jobs.

- A new job can start only after the completion of the previous job on the same line, i.e., pre-emption is not allowed.
- The assembly lines are unrelated, which means that the process time for each job depends on the assembly line to which the job is assigned. These process times can be obtained through solving a set of Component Allocation Problems (CAPs), which has been investigated in Chapter 3. Because the proposed solution method for the CAP has not been implemented for practical use in the investigated manufacturer, these time values can be obtained by using the vendor software of the machine vendor (See Subsection 3.6.2.2).
- For practical considerations, a back-side job (a job processing the back sides of boards) can only begin 2 hours after the start time of the corresponding front-side job.
- The PCBs are categorized into two types: those should meet the RoHS (Restriction of Hazardous Substances) compliance and those are not required to. A common setup time of 16 minutes (0.27 hours) is required (for uploading programs, adjusting component feeders, etc). A special setup time of 2 hours is required when an RoHS job is processed right after a non-RoHS job on the same line.

Although not all of the above assumptions and constraints are valid for other PCB manufacturing environments, they are quite common and thus should be considered in the scheduling problem investigated in this research.

4.2.2 A mathematical model

Consider n jobs $\{J_1, J_2, \ldots, J_n\}$ to be processed on K assembly lines. Each job $J_i \in \{J_1, J_2, \ldots, J_n\}$ has a time window (a_i, b_i), where a_i is the earliest time that the job can begin (job ready time) and b_i is the latest time that the job should finish (job due date). For an assembly line k, there is also an earliest start time defined by R_k.

There is a nonnegative time cost for each line and each job s_{ik}, representing the process time for job i processed by line k, and a nonnegative time cost for each pair of jobs t_{ij} $(i, j \neq 0)$, representing the setup time (setup time) for processing job i immediately after job j on the same line. There is also a setup time $\bar{t}_{ki}$ required for line k to process the first job J_i $(i \neq 0)$ on that line.

In order to facilitate the formulation of the mathematical model, a dummy job J_0 that each assembly line starts with and a dummy job J_{n+1} that each assembly line finishes with are introduced. There is no time window for J_0 and J_{n+1}, and the process times for them are zero.

The notation used in the mathematical model is summarized as follows:

Sets and Indices:

i, j:	indices of jobs
k, k_1, k_2:	indices of assembly lines
N:	the set of all jobs excluding the dummy jobs (J_0 and J_{n+1})
N^0:	the set of all jobs including the dummy jobs (J_0 and J_{n+1})
N^1:	the set of jobs that process the front sides of boards

K:	the set of assembly lines
$B(i)$:	the job that processes the same boards but different sides with job i

Parameters:

n:	number of jobs
q:	weight for minimizing the makespan of all job*s*
p_i:	weight for minimizing tardiness of job i
s_{ik}:	process time for job i on line k
t_{ij}:	setup time for job j if it is processed right after job i on the same line
$\bar{t}_{ki}$:	setup time for line k if job i is the first job on line k, the value depends on the initial RoHS status of line k, i.e., the RoHS status of the job produced on the line before this planning period
a_i:	ready time for job i
b_i:	due date for job i
R_k:	ready time for line k

$M_1 \sim M_5$,: large positive constants

Decision variables:

x_{ijk}:	0-1 variable. = 1 if job j is processed right after i on line k, and = 0 otherwise
w_{ik}:	start time for job i on line k
C_{max}:	max completion time of the jobs, i.e., makespan
L_i:	tardiness (in time) for job i

The objective of the multi-line scheduling problem is to minimize the sum of weighted tardiness and weighted makespan, which is shown in the objective function (4.1).

$$\min(\sum_{i \in N} p_i L_i + qC_{\max}) \tag{4.1}$$

In the objective function, p_i is the weight for minimizing the tardiness of job i, which relates to the importance of the corresponding customer. q is the weight for minimizing the makespan C_{max}. In the model, p_i (i = 1, 2, …, n) are set to values much larger than q so that the model gives a higher priority to ensuring due date satisfaction.

The tardiness of each job, L_i, is defined by constraint sets (4.2) and (4.3) as a nonnegative value. In constraint set (4.2), $w_{ik} + s_{ik}$ represents the finish time of job i on line k, and b_i is the due date for job i. The term $M_1(1 - x_{ijk})$ ensures that this requirement applies only when job i and job j are processed on line k consecutively.

$$L_i \geq w_{ik} + s_{ik} - b_i - M_1(1 - x_{ijk}) \quad \forall k \in K, \forall i \in N, \forall j \in N^0 \tag{4.2}$$

$$L_i \geq 0 \quad \forall i \in N \tag{4.3}$$

The makespan is the completion time for all the jobs. Therefore, in constraint set (4.4), the makespan is set to be greater than the finish time of each job. In the constraint set, $w_{ik} + s_{ik}$ represents the finish time of job i on line k. Similarly, the term $M_2(1 - x_{ijk})$ ensures that the requirement applies only when job i and job j are processed on line k consecutively.

$$C_{\max} \geq w_{ik} + s_{ik} - M_2(1 - x_{ijk}) \quad \forall k \in K, \forall i \in N, \forall j \in N^0 \tag{4.4}$$

To ensure a feasible solution, the following constraints are required for a feasible job-line assignment. Constraint set (4.5) ensures that each job should be processed by exactly one line; constraint set (4.6) ensures that the number of the first producing jobs on the lines must not exceed *K*; similarly, constraint set (4.7) ensures the number of the last producing jobs on the lines must not exceed *K*. Constraint set (4.8) ensures that the same line processes jobs one by one, i.e., if job *j* is assigned to line *k*, both its predecessor and successor must be processed by line *k*.

$$\sum_{k \in K} \sum_{j \in N^0 \setminus \{i\}} x_{ijk} = 1 \quad \forall i \in N \tag{4.5}$$

$$\sum_{j \in N^0 \setminus \{0\}} x_{0jk} \le 1 \quad \forall k \in K \tag{4.6}$$

$$\sum_{i \in N^0 \setminus \{n+1\}} x_{i,n+1,k} \le 1 \quad \forall k \in K \tag{4.7}$$

$$\sum_{i \in N^0 \setminus \{j\}} x_{ijk} - \sum_{i \in N^0 \setminus \{j\}} x_{jik} = 0 \quad \forall k \in K, \forall j \in N \tag{4.8}$$

Constraint set (4.9) and constraint set (4.10) are introduced to ensure the start times of any two consecutive jobs *i* and *j* on the same line to be strictly increasing. The term $M_3(1-x_{ijk})$ in constraint set (4.9) ensures that the relative requirement applies only if job *j* is processed right after *i* on line *k*; similarly, the term $M_4(1-x_{0ik})$ in constraint set (4.10) ensures that the relative requirement applies only if job *i* is the first job to be processed on line *k*.

$$w_{ik} + s_{ik} + t_{ij} - w_{jk} \le M_3(1-x_{ijk}) \quad \forall k \in K, \forall i, j \in N, i \ne j \tag{4.9}$$

$$w_{0k} + \bar{t}_{ki} - w_{ik} \le M_4(1-x_{0ik}) \quad \forall k \in K, \forall i \in N \tag{4.10}$$

The start time for each line cannot be earlier than its ready time for the line and this is ensured by constraint set (4.11).

$$w_{0k} \geq R_k \quad \forall k \in K \tag{4.11}$$

Similarly, each job cannot start before its ready time and this is ensured by constraint set (4.12).

$$w_{ik} \geq a_i \quad \forall k \in K, \forall i \in N \tag{4.12}$$

In order to ensure that the process of a back-side job can only start 2 hours after the start time of the corresponding front-side job, constraint set (4.13) is introduced. The term $M_5(2 - \sum_{j \in N^0 \backslash \{i\}} x_{ijk_1} - \sum_{j \in N^0 \backslash \{i\}} x_{B(i)jk_2})$ is to ensure the appropriate condition for the constraint to apply.

$$(w_{ik_1} + 2) - w_{B(i)k_2} \leq M_5(2 - \sum_{j \in N^0 \backslash \{i\}} x_{ijk_1} - \sum_{j \in N^0 \backslash \{i\}} x_{B(i)jk_2})$$

$$\forall k_1 \in K, \forall k_2 \in K, \forall i \in N^1 \tag{4.13}$$

After the above discussion, a complete mathematical model for the scheduling problem can be written as follows.

$$\min(\sum_{i \in N} p_i L_i + qC_{\max}) \tag{4.1}$$

Subject to:

$$L_i \geq w_{ik} + s_{ik} - b_i - M_1(1 - x_{ijk}) \quad \forall k \in K, \forall i \in N, \forall j \in N^0 \tag{4.2}$$

$$L_i \geq 0 \quad \forall i \in N \tag{4.3}$$

$$C_{\max} \geq w_{ik} + s_{ik} - M_2(1 - x_{ijk}) \quad \forall k \in K, \forall i \in N, \forall j \in N^0 \tag{4.4}$$

$$\sum_{k\in K}\sum_{j\in N^0\backslash\{i\}} x_{ijk} = 1 \quad \forall i \in N \tag{4.5}$$

$$\sum_{j\in N^0\backslash\{0\}} x_{0jk} = 1 \quad \forall k \in K \tag{4.6}$$

$$\sum_{i\in N^0\backslash\{n+1\}} x_{i,n+1,k} = 1 \quad \forall k \in K \tag{4.7}$$

$$\sum_{i\in N^0\backslash\{j\}} x_{ijk} - \sum_{i\in N^0\backslash\{j\}} x_{jik} = 0 \quad \forall k \in K, \forall j \in N \tag{4.8}$$

$$w_{ik} + s_{ik} + t_{ij} - w_{jk} \le M_3(1 - x_{ijk}) \quad \forall k \in K, \forall i, j \in N, i \ne j \tag{4.9}$$

$$w_{0k} + \bar{t}_{ki} - w_{ik} \le M_4(1 - x_{0ik}) \quad \forall k \in K, \forall i \in N \tag{4.10}$$

$$w_{0k} \ge R_k \quad \forall k \in K \tag{4.11}$$

$$w_{ik} \ge a_i \quad \forall k \in K, \forall i \in N \tag{4.12}$$

$$(w_{ik_1} + 2) - w_{B(i)k_2} \le M_5(2 - \sum_{j\in N^0\backslash\{i\}} x_{ijk_1} - \sum_{j\in N^0\backslash\{i\}} x_{B(i)jk_2})$$

$$\forall k_1 \in K, \forall k_2 \in K, \forall i \in N^1 \tag{4.13}$$

$$x_{ijk} \in \{0, 1\} \quad \forall k \in K, \forall i, j \in N^0 \tag{4.14}$$

Model 4-1

The established model (4-1) is a Mixed Integer Linear Programming (MILP) model, which can be solved by several existing LP solver packages like CPLEX. Based on observation, the established model is found to be much akin to the Parallel-Machine Scheduling Problem (PMSP) if each assembly line is considered as a single machine. Sotskov and Shaklevich [Sot95] proved that the identical parallel-machine scheduling problem with makespan minimization, which is a relatively easy type of

PMSP, is *NP-hard*. For the investigated multi-line scheduling problem in PCB assembly, the assembly lines are not identical but unrelated, in that the process times for a job on different lines are different. In this sense, the multi-line scheduling problem can be viewed as an unrelated PMSP, which is much more difficult than the identical PMSP.

Furthermore, the combined objective of due date satisfaction and makespan minimization, sequence-dependent setup times, and precedence constraints may greatly increase the complexity of the problem.

4.2.3 Exact solutions

In order to verify the established model for the multi-line scheduling problem, some problem instances are generated and solved using a commercial LP solver, CPLEX 10.2.

In the experiments, seven problem instances are generated. For each instance, there are different numbers of jobs and different numbers of assembly lines, as shown in Table 4.1.

For each instance, the process time for a job on a particular line is determined randomly. In addition, there may be the case that a job cannot be processed by a particular line. In order to reflect real situations, the process time for job i on line k is generated as follows. The possibility that job i cannot be processed by line k is 0.2. If job i can be processed by line k, then the process time s_{ik} is generated randomly within [$avg_i - 1$, $avg_i + 1$] hours, where avg_i is an average process time for job i and is randomly generated within [3, 10] hours.

Table 4.1 Characteristics of test instances

Instance Name	Number of jobs	Number of lines
Test-n10k3	10	3
Test-n10k4	10	4
Test-n11k 3	11	3
Test-n11k4	11	4
Test-n12k4	12	4
Test-n15k4	15	4
Test-n20k4	20	4

The ready time for job *i*, i.e., a_i, is generated as follows. The possibility that the ready time for job *i* is 0, is set to 0.5. If the ready time for job *i* is not 0, then it is generated randomly within [0, $C_{\exp}$ / 2] hours, with the value rounded to an integer, where $C_{\exp}$ is a value which is computed by equation (4.15).

$$C_{\exp} = \left(\sum_{i=1}^{n} avg_i \right) / k \qquad (4.15)$$

The due date for job *i*, i.e., b_i, is generated randomly among [$C_{\exp}$ / 2, 2$C_{\exp}$], with the value rounded to an integer. The ready times and due dates are generated in this way to ensure a realistic number of jobs which have tight due dates. Whether a job processes single-sided PCBs, front-sides of PCBs, or back-sides of PCBs is determined randomly.

Table 4.2 shows the job data for the first instance Test-n10k3. The data for the other six instances are shown in Appendix II ～ VII.

Table 4.2 Data for the first instance Test-n10k3

job number	ready time [a]	due date [b]	*if_front* [c]	*bk_job* [d]	*RoHS* [e]	p_i [f]	process time [g]		
							line 1	line 2	line 3
1	0	15	0	\	0	2	4.56	5.02	5.34
2	0	9	0	\	1	2	6.34	6.01	6.98
3	4	21	1	4	0	3	1000	4.54	4.23
4	0	18	0	\	0	3	6.32	7.42	7.92
5	0	25	1	6	1	1	8.92	9.21	8.97
6	0	19	0	\	1	1	4.30	1000	1000
7	0	28	1	8	0	1	3.21	3.29	3.87
8	8	25	0	\	0	1	5.72	5.03	4.78
9	8	18	1	10	1	2	1000	8.92	9.87
10	10	27	0	\	1	2	7.72	6.92	7.34
						ready time of line	1.48	0	0.78
						initial RoHS status of lines[h]	1	0	0

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

All the time values in Table 4.2 are shown in hours. If a job cannot be processed on an assembly line, then the process time is set to be 1000. The weights for tardiness penalty, p_i , are randomly chosen from {1, 2, 3}, with a higher value representing greater importance of the customer. The weight for makespan penalty, q, is set to a much smaller value, 0.01, so that the problem gives a higher priority to tardiness minimization.

With the data in Table 4.2, the complete MILP model for instance Test-n10k3 can be obtained. Note that the large numbers in the Model, M_1~M_5, are all set to 10000. The model is input into CPLEX. The complete model is listed as follows.

Minimize

2.00L01+2.00L02+3.00L03+3.00L04+1.00L05+1.00L06+1.00L07+1.00L08 +2.00L09+2.00L10+0.01Cmax

Subject to

Constraint set (4.2):

cons.1:	L01 - w0101 - 10000x010201 > = - 10010.44
cons.2:	L01 - w0101 - 10000x010301 > = - 10010.44
cons.3:	L01 - w0101 - 10000x010401 > = - 10010.44
cons.4:	L01 - w0101 - 10000x010501 > = - 10010.44
cons.5:	L01 - w0101 - 10000x010601 > = - 10010.44
cons.6:	L01 - w0101 - 10000x010701 > = - 10010.44
cons.7:	L01 - w0101 - 10000x010801 > = - 10010.44
cons.8:	L01 - w0101 - 10000x010901 > = - 10010.44
cons.9:	L01 - w0101 - 10000x011001 > = - 10010.44
cons.10:	L01 - w0101 - 10000x011101 > = - 10010.44
cons.11:	L02 - w0201 - 10000x020101 > = - 10002.66

……

cons.300: L10 - w1003 - 10000x101103 > = - 10019.66

Constraint set (4.3):

cons.301: L01 > = 0

cons.302: L02 > = 0

cons.303: L03 > = 0

cons.304: L04 > = 0

cons.305: L05 > = 0

cons.306: L06 > = 0

cons.307: L07 > = 0

cons.308: L08 > = 0

cons.309: L09 > = 0

cons.310: L10 > = 0

Constraint set (4.4):

cons.311: Cmax - w0101 - 10000x010201 > = - 9995.44

cons.312: Cmax - w0101 - 10000x010301 > = - 9995.44

cons.313: Cmax - w0101 - 10000x010401 > = - 9995.44

cons.314: Cmax - w0101 - 10000x010501 > = - 9995.44

cons.315: Cmax - w0101 - 10000x010601 > = - 9995.44

cons.316: Cmax - w0101 - 10000x010701 > = - 9995.44

cons.317: Cmax - w0101 - 10000x010801 > = - 9995.44

cons.318: Cmax - w0101 - 10000x010901 > = - 9995.44

cons.319: Cmax - w0101 - 10000x011001 > = - 9995.44

cons.320: Cmax - w0101 - 10000x011101 > = - 9995.44

cons.321: Cmax - w0201 - 10000x020101 > = - 9993.66

……

cons.610: Cmax - w1003 - 10000x101103 > = - 9992.66

Constraint set (4.5):

cons.611: x010201 + x010202 + x010203 + x010301 + x010302 + x010303 +

x010401 + x010402 + x010403 + x010501 + x010502 + x010503 + x010601 + x010602 + x010603 + x010701 + x010702 + x010703 + x010801 + x010802 + x010803 + x010901 + x010902 + x010903 + x011001 + x011002 + x011003 + x011101 + x011102 + x011103 = 1

cons.612: x020101 + x020102 + x020103 + x020301 + x020302 + x020303 + x020401 + x020402 + x020403 + x020501 + x020502 + x020503 + x020601 + x020602 + x020603 + x020701 + x020702 + x020703 + x020801 + x020802 + x020803 + x020901 + x020902 + x020903 + x021001 + x021002 + x021003 + x021101 + x021102 + x021103 = 1

cons.613: x030101 + x030102 + x030103 + x030201 + x030202 + x030203 + x030401 + x030402 + x030403 + x030501 + x030502 + x030503 + x030601 + x030602 + x030603 + x030701 + x030702 + x030703 + x030801 + x030802 + x030803 + x030901 + x030902 + x030903 + x031001 + x031002 + x031003 + x031101 + x031102 + x031103 = 1

……

cons.620: x100101 + x100102 + x100103 + x100201 + x100202 + x100203 + x100301 + x100302 + x100303 + x100401 + x100402 + x100403 + x100501 + x100502 + x100503 + x100601 + x100602 + x100603 + x100701 + x100702 + x100703 + x100801 + x100802 + x100803 + x100901 + x100902 + x100903 + x101101 + x101102 + x101103 = 1

Constraint set (4.6):

cons.621: x000101 + x000201 + x000301 + x000401 + x000501 + x000601 + x000701 + x000801 + x000901 + x001001 + x001101 = 1

cons.622: x000102 + x000202 + x000302 + x000402 + x000502 + x000602 + x000702 + x000802 + x000902 + x001002 + x001102 = 1

cons.623: x000103 + x000203 + x000303 + x000403 + x000503 + x000603 + x000703 + x000803 + x000903 + x001003 + x001103 = 1

Constraint set (4.7):

cons.624: x001101 + x011101 + x021101 + x031101 + x041101 + x051101 +

x061101 + x071101 + x081101 + x091101 + x101101 = 1

cons.625: x001102 + x011102 + x021102 + x031102 + x041102 + x051102 + x061102 + x071102 + x081102 + x091102 + x101102 = 1

cons.626: x001103 + x011103 + x021103 + x031103 + x041103 + x051103 + x061103 + x071103 + x081103 + x091103 + x101103 = 1

Constraint set (4.8):

cons.627: x000101 + x020101 + x030101 + x040101 + x050101 + x060101 + x070101 + x080101 + x090101 + x100101 - x010201 - x010301 - x010401 - x010501 - x010601 - x010701 - x010801 - x010901 - x011001 - x011101 = 0

cons.628: x000201 + x010201 + x030201 + x040201 + x050201 + x060201 + x070201 + x080201 + x090201 + x100201 - x020101 - x020301 - x020401 - x020501 - x020601 - x020701 - x020801 - x020901 - x021001 - x021101 = 0

cons.629: x000301 + x010301 + x020301 + x040301 + x050301 + x060301 + x070301 + x080301 + x090301 + x100301 - x030101 - x030201 - x030401 - x030501 - x030601 - x030701 - x030801 - x030901 - x031001 - x031101 = 0

……

cons.656: x001003 + x011003 + x021003 + x031003 + x041003 + x051003 + x061003 + x071003 + x081003 + x091003 - x100103 - x100203 - x100303 - x100403 - x100503 - x100603 - x100703 - x100803 - x100903 - x101103 = 0

Constraint set (4.9)

cons.657: w0101 - w0201 + 10000x010201 < = 9993.44

cons.658: w0101 - w0301 + 10000x010301 < = 9995.17

cons.659: w0101 - w0401 + 10000x010401 < = 9995.17

cons.660: w0101 - w0501 + 10000x010501 < = 9993.44

cons.661: w0101 - w0601 + 10000x010601 < = 9993.44

cons.662: w0101 - w0701 + 10000x010701 < = 9995.17

cons.663: w0101 - w0801 + 10000x010801 < = 9995.17

cons.664: w0101 - w0901 + 10000x010901 < = 9993.44

cons.665: w0101 - w1001 + 10000x011001 < = 9993.44

cons.666: w0201 - w0101 + 10000x020101 < = 9993.39

……

cons.926: w1003 - w0903 + 10000x100903 < = 9992.39

Constraint set (4.10):

cons.927: w0001 - w0101 + 10000x000101 < = 9999.73

cons.928: w0001 - w0201 + 10000x000201 < = 9999.73

cons.929: w0001 - w0301 + 10000x000301 < = 9999.73

cons.930: w0001 - w0401 + 10000x000401 < = 9999.73

cons.931: w0001 - w0501 + 10000x000501 < = 9999.73

cons.932: w0001 - w0601 + 10000x000601 < = 9999.73

cons.933: w0001 - w0701 + 10000x000701 < = 9999.73

cons.934: w0001 - w0801 + 10000x000801 < = 9999.73

cons.935: w0001 - w0901 + 10000x000901 < = 9999.73

cons.936: w0001 - w1001 + 10000x001001 < = 9999.73

cons.937: w0002 - w0102 + 10000x000102 < = 9999.73

……

cons.956: w0003 - w1003 + 10000x001003 < = 9998.00

Constraint set (4.11):

cons.957: w0001 > = 1.48

cons.958: w0002 > = 0.00

cons.959: w0003 > = 0.78

Constraint set (4.12):

cons.960: w0101 > = 0.00

cons.961: w0201 > = 0.00

cons.962: w0301 > = 4.00

cons.963: w0401 > = 0.00

cons.964: w0501 > = 0.00

cons.965: w0601 > = 0.00

cons.966: w0701 > = 0.00

cons.967: w0801 > = 8.00

cons.968: w0901 > = 8.00

cons.969: w1001 > = 10.00

cons.970: w0102 > = 0.00

……

cons.989: w1003 > = 10.00

Constraint set (4.13):

cons.990: w0301 - w0401 + 10000x030101 + 10000x030201 + 10000x030401 + 10000x030501 + 10000x030601 + 10000x030701 + 10000x030801 + 10000x030901 + 10000x031001 + 10000x031101 + 10000x040101 + 10000x040201 + 10000x040301 + 10000x040501 + 10000x040601 + 10000x040701 + 10000x040801 + 10000x040901 + 10000x041001 + 10000x041101 < = 19998

cons.991: w0501 - w0601 + 10000x050101 + 10000x050201 + 10000x050301 + 10000x050401 + 10000x050601 + 10000x050701 + 10000x050801 + 10000x050901 + 10000x051001 + 10000x051101 + 10000x060101 + 10000x060201 + 10000x060301 + 10000x060401 + 10000x060501 + 10000x060701 + 10000x060801 + 10000x060901 + 10000x061001 + 10000x061101 < = 19998

cons.992: w0701 - w0801 + 10000x070101 + 10000x070201 + 10000x070301 + 10000x070401 + 10000x070501 + 10000x070601 + 10000x070801 + 10000x070901 + 10000x071001 + 10000x071101 + 10000x080101 + 10000x080201 + 10000x080301 + 10000x080401 + 10000x080501 + 10000x080601 + 10000x080701 + 10000x080901 + 10000x081001 + 10000x081101 < = 19998

```
cons.993:    w0901 - w1001 + 10000x090101 + 10000x090201 + 10000x090301 +
             10000x090401 + 10000x090501 + 10000x090601 + 10000x090701 +
             10000x090801 + 10000x091001 + 10000x091101 + 10000x100101 +
             10000x100201 + 10000x100301 + 10000x100401 + 10000x100501 +
             10000x100601 + 10000x100701 + 10000x100801 + 10000x100901 +
             10000x101101 < = 19998
cons.994:    w0301 - w0402 + 10000x030101 + 10000x030201 + 10000x030401 +
             10000x030501 + 10000x030601 + 10000x030701 + 10000x030801 +
             10000x030901 + 10000x031001 + 10000x031101 + 10000x040102 +
             10000x040202 + 10000x040302 + 10000x040502 + 10000x040602 +
             10000x040702 + 10000x040802 + 10000x040902 + 10000x041002 +
             10000x041102 < = 19998
……
cons.1025:   w0903 - w1003 + 10000x090103 + 10000x090203 + 10000x090303 +
             10000x090403 + 10000x090503 + 10000x090603 + 10000x090703 +
             10000x090803 + 10000x091003 + 10000x091103 + 10000x100103 +
             10000x100203 + 10000x100303 + 10000x100403 + 10000x100503 +
             10000x100603 + 10000x100703 + 10000x100803 + 10000x100903 +
             10000x101103 < = 19998

Bounds
Cmax   free
L01    free          L02    free          L03    free          L04    free
L05    free          L06    free          L07    free          L08    free
L09    free          L10    free
w0001 free           w0002 free           w0003 free           w0101 free
w0102 free           w0103 free           w0201 free           w0202 free
w0203 free           w0301 free           w0302 free           w0303 free
w0401 free           w0402 free           w0403 free           w0501 free
w0502 free           w0503 free           w0601 free           w0602 free
w0603 free           w0701 free           w0702 free           w0703 free
```

```
w0801 free          w0802 free          w0803 free          w0901 free
w0902 free          w0903 free          w1001 free          w1002 free
w1003 free

Binary
x000101     x000201     x000301     x000401     x000501     x000601
x000701     x000801     x000901     x001001     x001101     x010101
x010201     x010301     x010401     x010501     x010601     x010701
x010801     x010901     x011001     x011101     x020101     x020301
x020401     x020501     x020601     x020701     x020801     x020901
x021001     x021101     x030101     x030201     x030301     x030401
x030501     x030601     x030701     x030801     x030901     x031001
……
x100603     x100703     x100803     x100903     x101003     x101103

End
```

The test instances are solved by CPLEX 10.2, running on a desktop machine with Intel Xeon 2.80 GHz CPU and 2 GB RAM. Of the seven instances, only four instances can be solved to optimality, and the computational times are great.

The CPLEX results and optimal solution for the first instance Test-n10k3 are listed in the following. Table 4.3 summarizes the characteristics of the model and the optimal solution for the instance. For this instance with only 10 jobs and 3 assembly lines, there are totally 377 variables (333 integer variables) and 1025 linear constraints. CPLEX spent 1441 seconds to solve the problem to optimality. The CPLEX results and optimal solutions for Test-n10k4, Test-n11k3, and Test-n11k4 are given in Appendices II, III, and IV, respectively.

```
Integer optimal
Objective =   6.5810000000e-001
Solution time = 1441 sec.
Iterations = 19819576
Nodes = 823593
```

Variable Name	Solution Value	Variable Name	Solution Value
L06	0.200000	L08	0.190000
Cmax	26.81		
x000101	1.000000	x010401	1.000000
x040601	1.000000	x060801	1.000000
x081101	1.000000	x000202	1.000000
x020902	1.000000	x090702	1.000000
x071102	1.000000	x000303	1.000000
x030503	1.000000	x051003	1.000000
x101103	1.000000		
w0101	1.75	w0401	6.58
w0601	14.9	w0801	19.47
w0202	2	w0902	8.28
w0702	17.47	w0303	4
w0503	10.23	w1003	19.47

All other variables are zero.

Table 4.3 Characteristics of the model and optimal solution for Test-n10k3

No. of integer variables	No. of variables	No. of linear constraints	Objective (hours)	CPU time (sec.)
333	377	1025	0.6581	1441

The Gantt chart in Figure 4.1 shows the optimal schedule for instance Test-n10k3. The start times for the jobs are shown above the job items. For example, job 1, job 4, job 6, and job 8 are processed on line 1, consecutively. Because the ready time for line 2 is 1.48 and the initial status of line 1 is RoHS (as shown in Table 4.2), the start time for job 1 on line 1, w_{11} = 1.48 + 0.27 = 1.75 (0.27 is the normal setup time). Similarly, because both job 1 and job 4 are non-RoHS jobs, the setup time between them is also 0.27. The start time for job 4 on line 1, $w_{41} = w_1 + s_{11} + t_{14}$ = 1.75 + 4.56 + 0.27 = 6.58. However, the setup time between Job 1 and Job 6 is 2 hours because Job 1 is a non-RoHS Job and Job 6 is an RoHS job. Therefore, the start time for job 6 on line 1, $w_{61} = w_4 + s_{41} + t_{46}$ = 6.58 + 6.32 + 2 = 14.9.

From the figure, it can be seen that the precedence constraints between front-side jobs and back-side jobs are also satisfied. For example, job 7 and job 8 are the front sides and back sides of the same boards, respectively. Therefore, the start time of job 8 cannot be earlier than 2 hours after the start time of job 7.

Based on the obtained schedule, there are two jobs that are completed beyond their due dates, i.e., job 6 and job 8. For job 6, the complete time is 19.2 and the due date is 19. For job 8, the complete time is 25.19 and the due date is 25. The makespan for all the jobs is 26.81, which is the complete time for the last finished job, job 10. The objective value for this optimal solution can be calculated as (19.2 − 19) + (25.19 − 25) + 0.01×26.81 = 0.6581.

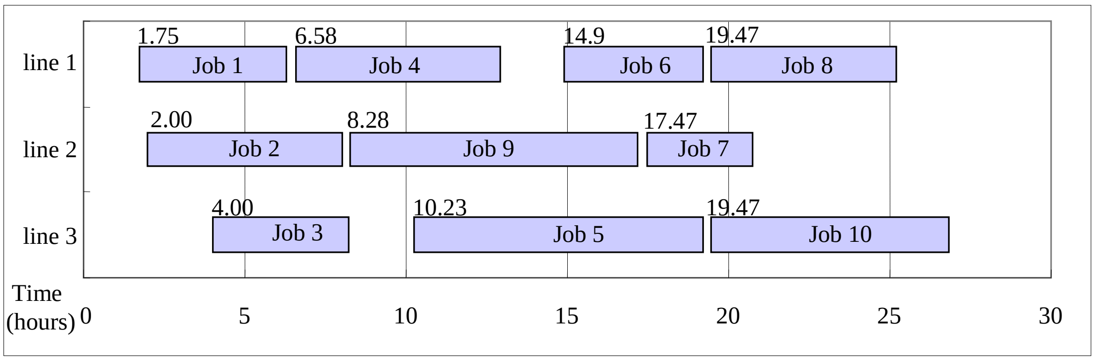


Figure 4.1 A Gantt chart for the optimal schedule of the instance Test-n10k3

The results for all the seven instances are summarized in Table 4.4. Due to the great complexity, CPLEX only obtains the optimal solutions to the first four instances and the computation times are substantial. When the problem size increases, the complexity of the MILP model increases exponentially. For the instance Test-n12k4, which has two more jobs and one more line than the instance Test-n10k3, the number of variables and the number of constraints are almost doubled. For Test-n12k4, CPLEX fails to find the optimal solution within 10 days.

For a realistic problem instance, there may be many more jobs and assembly lines than the test instances. Thus, optimal solutions are infeasible for realistic MLSP instances. For this reason, a more efficient heuristic solution is required.

Table 4.4 CPLEX computational results for the test instances

Instances	No. of variables	No. of linear constraints	Optimal solution	CPU time (minutes)
Test-n10k3	377	1025	0.6581	24.0
Test-n10k4	499	1376	0.2311	33.0
Test-n11k3	447	1213	2.1005	161.2
Test-n11k4	592	1638	8.1449	1669.4
Test-n12k4	693	1924	No solution found for 240 hours	
Test-n15k4	1044	2910	No solution found for 240 hours	
Test-n20k4	1789	5076	No solution found for 240 hours	

4.3 A GENETIC ALGORITHM FOR THE MLSP

As discussed in the previous section, the multi-line scheduling problem is very complex for exact solutions. In order to solve the problem efficiently, a specific Genetic Algorithm (GA) is proposed.

4.3.1 Representation scheme

For the multi-line scheduling problem, the solution representation is not straightforward, because a solution to the problem consists of assigning jobs to assembly lines and sequencing jobs within each line at the same time.

In the proposed genetic algorithm, a special permutation-type representation scheme is suggested (see Figure 4.2). For a problem with n jobs and k lines, the chromosome for a solution x has the form of a string of length (n + k). Figure 4.2

illustrates a chromosome for the problem instance Test-n10k3, which is discussed in Section 4.2. The instance has 10 jobs and 3 assembly lines. In the chromosome shown in the figure, the genes valued 1 to 10 represent jobs 1 to 10, respectively, while genes valued 11, 12, and 13 represent lines 1, 2, and 3, respectively.

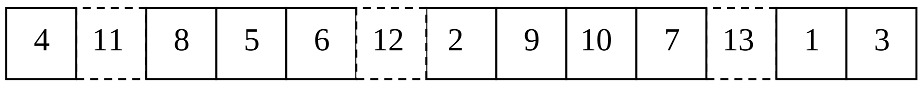


Figure 4.2 A chromosome in the proposed GA for the MLSP

Based on the chromosome in Figure 4.2, jobs 8, 5 and 6 are processed by line 1 consecutively. Similarly, jobs 2, 9, 10 and 7 are processed by line 2 consecutively, and jobs 1, 3 and 4 are processed by line 3 consecutively.

The scheduling solution represented by the chromosome in Figure 4.2 is shown as follows:

Line 1:	Job 8→Job 5→Job 6
Line 2:	Job 2→Job 9→Job 10→Job 7
Line 3:	Job 1→Job 3→Job 4

The advantage of using such a permutation-type representation scheme is that many genetic operators for permutation representation are available and can be readily used in the proposed GA, which will be discussed later.

4.3.2 Fitness evaluation

Since the investigated problem is a minimization problem, the fitness value of a chromosome is set to be the inverse of the objective value, which is defined by function (4.1) in Model 4-1.

In order to compute the objective value for a chromosome, the start times for all the jobs need to be determined according to the chromosome representation. The following illustrates the procedure for the determination of the job start times according to a given chromosome.

Step 1: For each line in sequence, consider the start time for the first job. If the start time for the first job can be determined, move on to determine the start times for the following jobs on the line one by one, until the start time of a certain job cannot be determined.

Step 2: For each line in sequence, reconsider the start time for the job that cannot be determined in Step 1.

Step 3: Repeat Step 2 until the start times for all the jobs have been determined, or no start time of any job can be determined.

Because the objective of the scheduling problem considers due date satisfaction and makespan reduction, both of which require reducing the finish times for the jobs, the start time for each job should be as early as possible. For a non-back-side job, the start time should consider the finish time for the previous job and its own ready time. For a back-side job, however, the special precedence constraint should also be considered, that is, a back-side job can only begin 2 hours after the start time of the corresponding front-side job (see Section 4.2). For example, suppose that job *i* and job *j* process the front sides and back sides of the same boards, respectively, and are produced on the same line consecutively. If the start time and process time for job *i* is *a* and *b*, respectively, the setup time from job *i* to job *j* is *c*, and the ready time for job *j* is *d*, then the start time for job *j* should be $max\ \{a + b + c, a + 2, d\}$.

Consider the chromosome in Figure 4.2, which represents a solution to the instance Test-n10k3. The procedure for deciding the job start times can be described as follows.

1. On Line 1, Job 8 is a back-side job and the start time of its corresponding front-side job (Job 7) has not yet been determined. Therefore, the procedure moves on to consider Line 2.
2. On Line 2, the ready time for the line is 0. Because the initial status for Line 2 is non-RoHS and the first job, i.e., Job 2, is an RoHS job. So the setup time of Job 2 is 2 hours. The start time of Job 2 is 0 + 2 = 2 hours. The process time for Job 2 on Line 1 is 6.01 hours. So the complete time of Job 2 is 2 + 6.01 = 8.01 hours. The next job on Line 2 is Job 9, which requires a setup time of 0.27 hours (because both Job 2 and Job 9 are RoHS jobs). So the start time of Job 9 is 8.01 + 0.27 = 8.28 hours. The process time for Job 9 on Line 2 is 8.92 hours. So the complete time of Job 9 is 8.28 + 8.92 = 17.2 hours. The next job on Line 2 is Job 10, which requires a setup time of 0.27 hours (because both Job 9 and Job 10 are RoHS jobs). So the start time of Job 10 is 17.2 + 0.27 = 17.47 hours. The process time for Job 10 on Line 2 is 6.92 hours. So the complete time of Job 10 is 17.47 + 6.92 = 24.39 hours. The last job on Line 2

is Job 7, which also requires a setup time of 0.27 hours. So the start time of Job 7 is 24.39 + 0.27 = 24.66 hours. The process time for Job 7 on Line 2 is 3.29 hours. So the complete time of Job 7 is 24.66 + 3.29 = 27.95 hours.

3. On Line 3, the ready time for the line is 0.78. Because the initial status for Line 3 is non-RoHS and the first job, i.e., Job 1, is a non-RoHS job. So the setup time of Job 2 is 0.27 hours. The start time of Job 1 is 0.78 + 0.27 = 1.05 hours. The process time for Job 1 on Line 3 is 5.34 hours. So the complete time of Job 1 is 1.05 + 5.34 = 6.39 hours. The next job on Line 3 is Job 3, which requires a setup time of 0.27 hours (because both Job 2 and Job 9 are non-RoHS jobs). So the start time of Job 3 is 6.39 + 0.27 = 6.66 hours. The process time for Job 3 on Line 3 is 4.23 hours. So the complete time of Job 5 is 6.66 + 4.23 = 10.89 hours. The last job on Line 3 is Job 4, which requires a setup time of 0.27 hours (because both Job 3 and Job 4 are non-RoHS jobs). So the start time of Job 4 is 10.89 + 0.27 = 11.16 hours. The process time for Job 4 on Line 3 is 7.92 hours. So the complete time of Job 4 is 11.16 + 7.92 = 19.08 hours.

4. The procedure moves back to consider the jobs on Line 1. For Job 8 which is a back-side job corresponding to Job 7, the start time of Job 8 can only be 2 hours later than the start time of Job 7. Therefore, the start time of Job 8 is 24.66 + 2 = 26.66. The process time for Job 8 on Line 1 is 5.72 hours. So the complete time of Job 8 is 26.66 + 5.72 = 32.38 hours. The next job on Line 1 is Job 5, which requires a setup time of 2 hours (because Job 8 is a non-RoHS job and Job 5 is an RoHS job). So the start time of Job 5 is 32.38 + 2 = 34.38

hours. The process time for Job 5 on Line 1 is 8.92 hours. So the complete time of Job 5 is 34.38 + 8.92 = 43.3 hours. The last job on Line 1 is Job 6, which requires a setup time of 0.27 hours (because both Job 5 and Job 6 are RoHS jobs). So the start time of Job 6 is 43.3 + 0.27 = 43.57 hours. The process time for Job 6 on Line 1 is 4.30 hours. So the complete time of Job 6 is 43.57 + 4.30 = 47.87 hours.

Following the above procedure, the detailed schedule represented by the GA chromosome in Figure 4.2 can be obtained, which is shown as the Gantt chart in Figure 4.3.

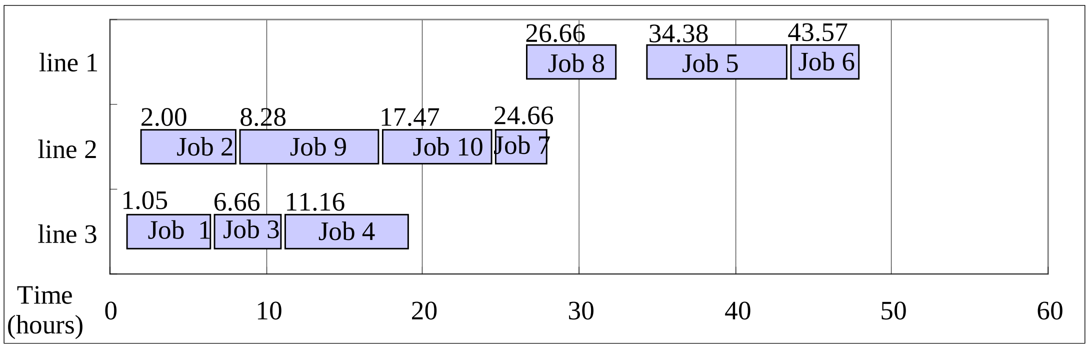


Figure 4.3 Detailed schedule represented by a chromosome in the GA

Based on the schedule shown in Figure 4.3, there are 4 jobs, i.e., Job 4, Job 5, Job 8, and Job 6, which are completed beyond due dates. The makespan for all the jobs is 47.87 hours, which is the completion time of Job 6. The objective value is then calculated using the objective function in Model 4-1.

$$Objective = \sum_{i \in N} p_i L_i + qC_{\max} = 3 \times (19.08 - 18) + 1 \times (43.3 - 25) + 1 \times (47.87 - 19) + 1 \times (32.38 - 25) + 0.01 \times 47.87 = 58.2687.$$

The fitness value of the chromosome is the inverse of the objective value:

$$Fitness = 1 / 58.2687 = 0.0172.$$

It should be noted that there may be some chromosomes that represent infeasible solutions. Consider the following scheduling solution for the same problem:

Line 1:	Job 4→Job 5→Job 1
Line 2:	Job 8→Job 9→Job 10→Job 7
Line 3:	Job 10→Job 3→Job 4

For the above solution, all the first jobs on the three lines, i.e., Job 4, Job 8 and Job 10, are the back-side jobs, which cannot be processed before their corresponding front-side jobs. In the genetic algorithm, the fitness values of such infeasible chromosomes are set to zero.

4.3.3 Genetic operators

Because the proposed GA adopts a permutation-type representation scheme, the genetic operators for a permutation-type representation can be readily used for the proposed GA, which include partially mapped crossover (PMX), edge crossover, order crossover (OX), cycle crossover, swap mutation, scramble mutation, inversion mutation, etc. In preliminary tests, it is found that there is no significant difference among the performance of the above crossover operators and the mutation operators for solving the investigated problem. In the proposed GA, the two commonly-used operators, the order crossover (OX) and swap mutation, will be used.

The swap mutation has been discussed in Chapter 3. The order crossover (OX) operator was designed by Davis [Dav91] and is suitable especially for permutation problems. The process of the operator can be described as follows.

1. Choose two crossover points at random, and copy the segment between them from the first parent into the first offspring.
2. Start from the second crossover point in the second parent, copy the remaining unused numbers into the first child in the order that they appear in the second parent, wrapping around at the end of the list.
3. Create the second offspring in an analogous manner, with the parent roles reversed.

The procedure for the OX operator is shown in Figure 4.4.

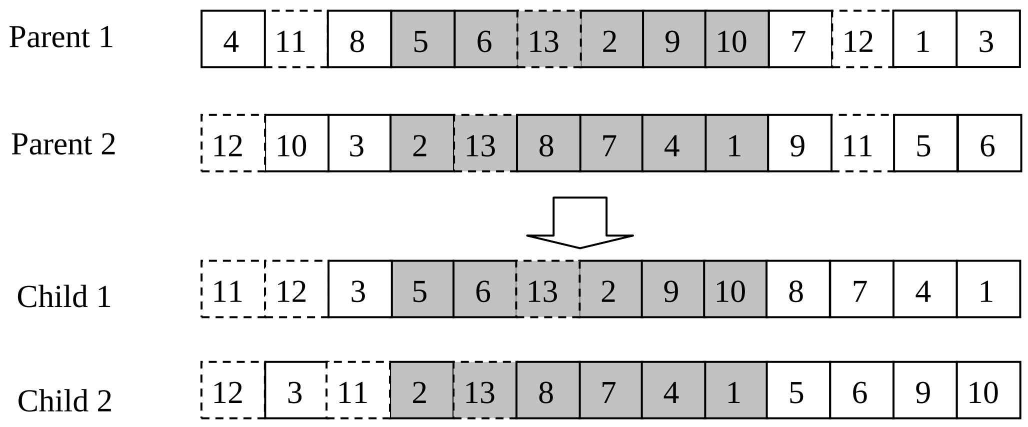


Figure 4.4 Order crossover (OX)

4.3.4 Replacement strategies

The replacement strategy defines how to select new individuals for the new generations. One of the commonly-used replacement strategies for genetic algorithms is the replace-worst strategy, which has been used in the genetic algorithm proposed for the component allocation problem (see Chapter 3).

However, due to the complicated structure of chromosome representation (which combines both the assignment of jobs to lines and sequencing of jobs in each line), the genetic operators may be ineffective for producing good offspring. Therefore, the replace-worst strategy may probably remove those potential chromosomes before they produce better offspring, and thus lead to early convergence. For this reason, a new replacement strategy is proposed and tried in the genetic algorithm proposed for the scheduling problem.

The new replacement strategy incorporates two different replacement policies, i.e., the replace-parent policy and the replace-worst policy. For the replace-parent policy, every new offspring only compares with its parents, and replaces the parent that is worse than the offspring. The replace-parent policy is applied in every generation while the replace-worst policy is applied only once during certain number of generations. By adjusting the frequency for the replace-worse policy, the convergence of the searching process and the diversity of the population can be well balanced.

In order to examine the effectiveness of the new replacement strategy, two GAs are implemented and compared in the following experimental tests, i.e., the GA-R, which uses the common replace-worst replacement strategy, and the GA-N, which uses the new replacement strategy. The general process for the GA-R is the same as the GA for the Component Allocation Problem (CAP) discussed in Chapter 3 (see Figure 3.7). The general process for the GA-N is illustrated in Figure 4.5. In the figure, C_w is the cycle (in generations) for applying the replace-worst policy. For example, if C_w = 20, then the replace-worst policy will apply once every 20 generations.

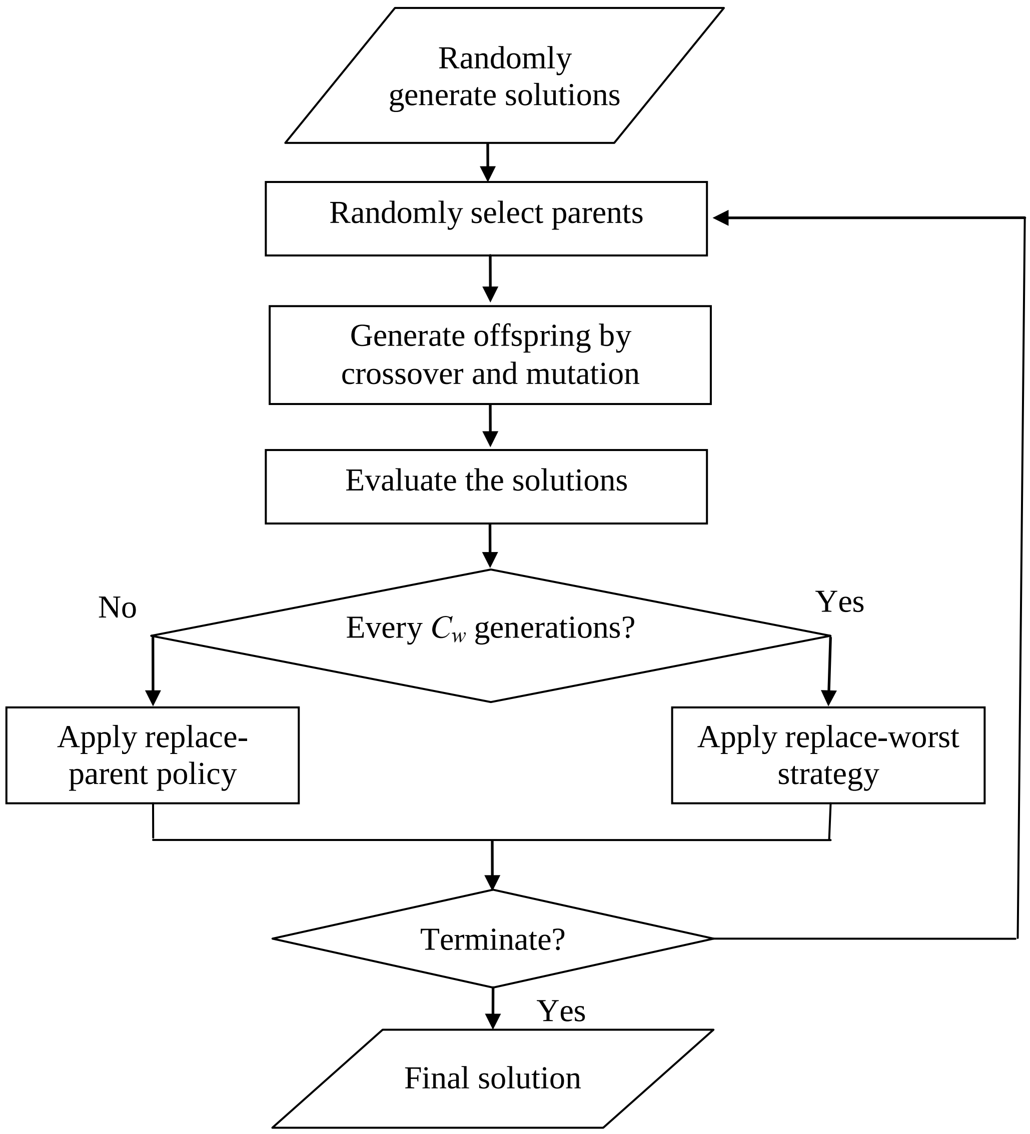

Randomly generate solutions
Randomly select parents
Generate offspring by crossover and mutation
Evaluate the solutions
No
Every C_w generations?
Yes
Apply replace-parent policy
Apply replace-worst strategy
Terminate?
Yes
Final solution

Figure 4.5 General framework for the GA-N for the MLSP

4.4 EXPERIMENTAL TESTS

The test instances generated in Section 4.2 are used to test the proposed genetic algorithms. The two GAs using two different replacement strategies, i.e., GA-R and GA-N, are tested.

The GAs are coded in Microsoft Visual C++ 2005, and run on a desktop machine with Intel Xeon 2.80 GHz CPU and 2 GB RAM, which is the same machine used by CPLEX in Section 4.2.

4.4.1 Parameter setting for the GAs

Like the GA for the Component Allocation Problem (CAP) in Chapter 3, the parameters for the genetic operators for the GA-R and GA-N for the multi-line scheduling problem are determined through trial and error. The crossover rate C_r is set to 0.9. That is, 90% of the population members are randomly selected as parents to produce 90 children in each generation. The swap mutation rate M_r is set to 0.1, which means that each child is selected with possibility of 0.1 to be mutated. Similar to the discussion in Chapter 3, the population sizes for both GAs are set to 100.

For the GA-N, it is found that the cycle for the replace-worst policy, C_w, has significant effect on the performance of the algorithm. In order to evaluate this effect and determine the most suitable value for C_w, the following experiments are conducted.

Figure 4.6 shows the results for the instance Test-n20k4 obtained by GA-N with different C_w values, i.e., 1, 5, 20, 60, and 200 generations. For each run, the algorithm terminates when there is no improvement during 3000 generations. In the figure, the unit for C_w values are shown in unit of n generations (n is the number of jobs, i.e., 20). From the results, it can be seen that the GA-N with C_w = 1 obtains a high objective value (which represent a bad solution). The objective value decreases rapidly while increasing the C_w value. It can be seen that the algorithm achieves the best objective value when C_w = 60 (i.e., 3n). When C_w > 60, the objective value increases again with the C_w value.

A similar observation is achieved for other instances. For example, Figure 4.7 shows the results for the instance Test-n15k4. It shows that the cycle for the replace-worst policy C_w has a significant effect on the performance of the GA-N. If C_w is too small, the replace-worst policy applies too frequently and thus a chromosome is not allowed to have enough time to generate good offspring. On the other hand, if C_w is too large, the algorithm spends too much time on exploitation that the exploration of the whole solution space is weak. In this sense, an appropriate C_w value is unique for achieving a good balance between exploitation and exploration of the algorithm.

From the experiments, it can be seen that the most appropriate C_w value is related to the problem size. For most of the problem instances, the appropriate C_w values range from 2n to 3n generations (n is the number of jobs). In the following experiments, C_w for the GA-N is set to 3n.

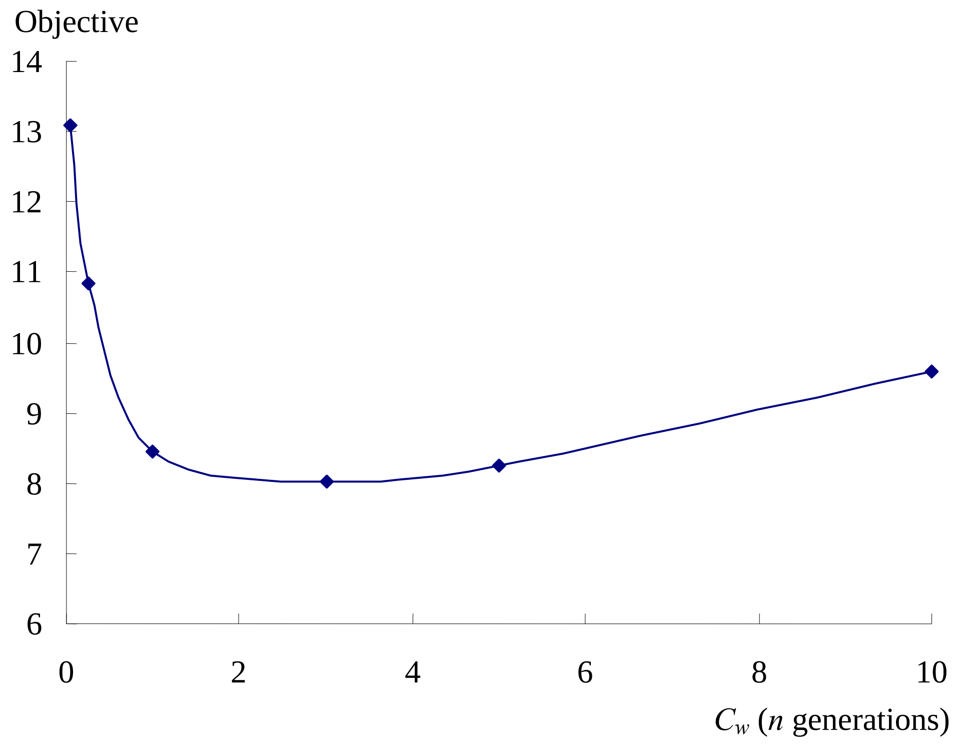


Figure 4.6 Objective values for Test-n20k4 obtained with different C_w values

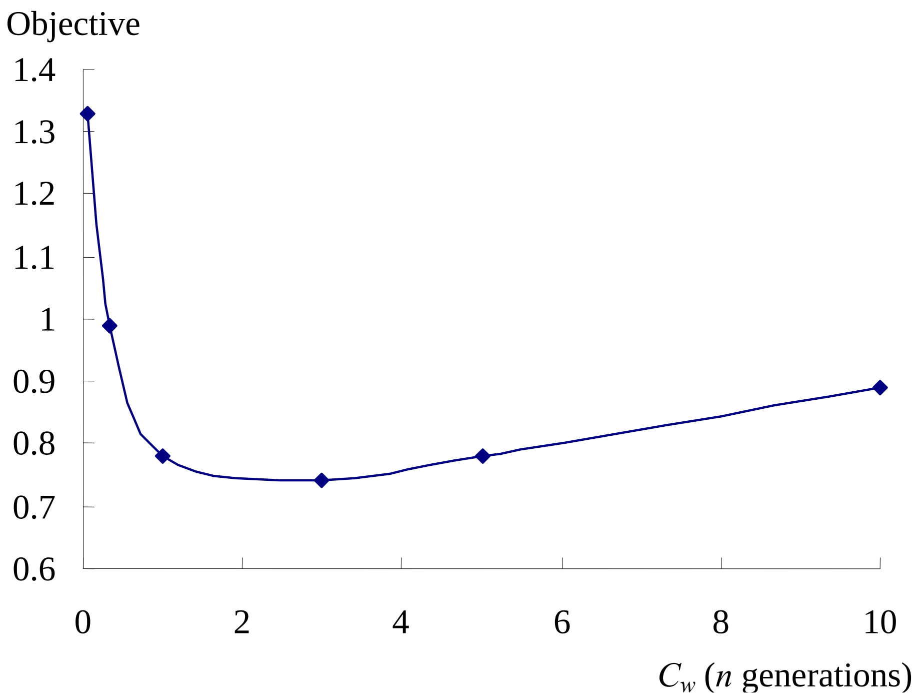


Figure 4.7 Objective values for Test-n15k4 obtained with different C_w values

4.4.2 Numerical results

The two GAs, i.e., the GA-R and GA-N, are run for all the generated instances. Both the algorithms are run 20 times for each instance. Each run terminates when there is no improvement during 3000 generations. Table 4.5 shows the results for all the problem instances obtained by GA-R and GA-N.

Table 4.5 Numerical results obtained by the two GAs

Instances	Optimal Values	GA-R			GA-N		
		ave. [a]	σ [b]	CPU [c]	ave. [a]	σ [b]	CPU [c]
Test-n10k3	0.6581	0.6581	0	3.84	0.6581	0	2.02
Test-n10k4	0.2311	0.2350	0.0082	4.27	0.2311	0	3.94
Test-n11k3	2.1005	2.4508	0.4628	3.15	2.1005	0	3.69
Test-n11k4	8.1449	8.3879	0.5433	4.45	8.1449	0	3.70
Test-n12k4	\	0.2439	0.0029	3.56	0.2439	0	3.66
Test-n15k4	\	1.5319	1.7071	3.91	0.7469	0.0151	3.90
Test-n20k4	\	14.3834	7.0809	6.22	8.1544	1.9878	7.05

[a] Average objective value for 20 runs;

[b] Standard deviation of objective values for 20 runs.

[c] Average CPU time (in seconds)

From table 4.5, it can be seen that the two GAs solve the problem instances with small computational times. The average objective values obtained by the GA-N are generally smaller than those obtained by the GA-R, while the computational times for the two GAs are similar. Furthermore, the GA-N achieves smaller standard deviation of objective values than the GA-R, indicating that the GA-N has greater reliability than the GA-R. For example, the objective value for the instance Test-n20k4 obtained by the GA-N is 8.1544, which is 43.3% smaller than that achieved by the GA-R, 14.3834. The standard deviation of objective values for the GA-N is 1.9878, which is also much smaller than that for the GA-R, 7.0809. The results show that the new replacement strategy significantly improves the performance of the genetic algorithm for solving the investigated problem.

Figure 4.8 shows the convergence process for the GA-R and the GA-N for the instance Test-n20k4. From the figure, it can be seen that the GA-N avoids early convergence that occurs in the GA-R and obtains much better solution in the end, This indicates that the exploration and exploitation of the algorithm are well balanced by adopting the new replacement strategy.

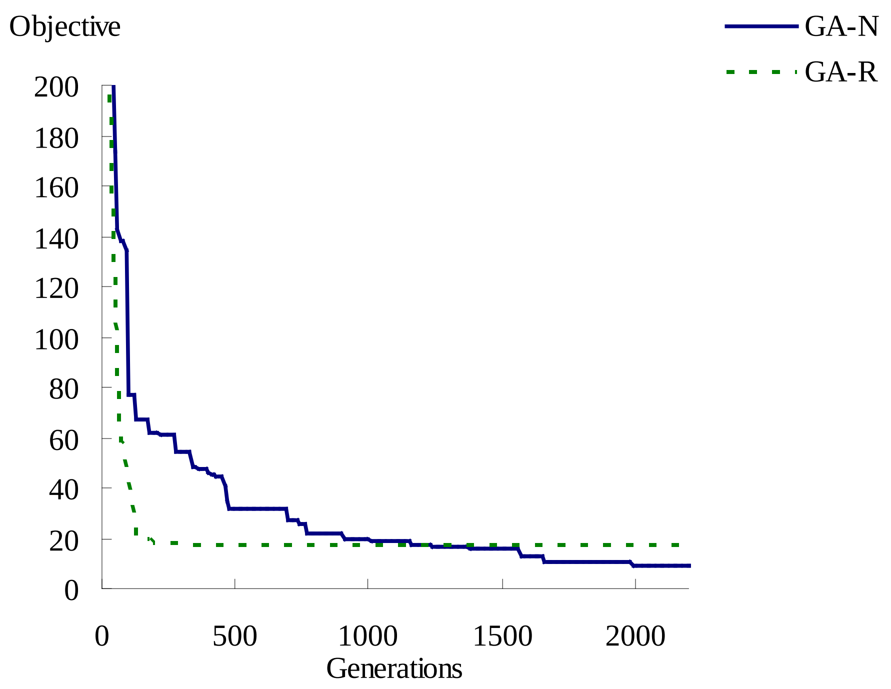


Figure 4.8 Convergence process for the GA-R and the GA-N

The GA-N obtains optimal solutions for all the four instances that can be solved by CPLEX, while the computational times for the GA-N are much shorter. For example, CPLEX spends 1669.4 minutes to solve the instance Test-n11k4 to optimality, while the GA-N spends only 3.70 seconds for obtaining the same results. Even for some large instances, which CPLEX fails to solve, the GA-N spend very short computational times for obtaining good results (The GA solutions for Test-n12k4, Test-n15k4, and Test-n20k4 are shown in Appendices V, VI, and VII). Even for the largest instance Test-n20k4, the GA-N obtains the result within 10 seconds, indicating that the GA-N is able to solve the problem efficiently.

4.5 A CASE STUDY

In order to evaluate the practical usefulness of the proposed GA method for the multi-line scheduling problem, a case study is conducted.

A realistic scheduling problem in an investigated PCB manufacturer is considered. Table 4.6 shows the job data for the problem. In this case, the jobs that are due within seven days are considered for the scheduling problem. There are totally 46 jobs to be processed by 5 assembly lines. The ready times and initial RoHS status of the assembly lines are shown in Table 4.7. The time values in both tables are shown in hours.

Table 4.6 Data for the problem in the case study

Job #	Ready time[a]	Due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i[f]	Process time[g]				
							Line 1	Line 2	Line 3	Line 4	Line 5
1	0	45	0	\	0	1	1000	11	11	1000	1000
2	0	45	0	\	0	1	13.62	13.2	13.2	12.75	12.75
3	45	245	1	4	0	1	15.89	14.49	14.49	13.87	13.87
4	45	245	0	\	0	1	15.89	14.49	14.49	14.64	14.64
5	0	245	1	6	1	1	17.48	15.12	15.12	15.12	15.12
6	0	245	0	\	1	1	23.64	18.89	18.89	26.31	26.31
7	0	245	1	8	1	1	24.51	22.75	24.68	22.75	22.75
8	0	245	0	\	1	1	24.51	22.75	24.68	22.75	22.75
9	22	178	1	10	1	1	15.23	12.75	15.14	12.75	14.75
10	22	178	0	\	1	1	15.23	12.75	15.14	12.75	14.75
11	0	89	0	\	1	1	8.89	6.5	7.52	1000	1000
12	45	178	1	13	0	1	14.23	11.89	1000	1000	1000
13	45	178	0	\	0	1	14.23	11.89	1000	1000	1000
14	45	201	1	15	0	1	12.5	12.22	11.73	11.78	11.78
15	45	201	0	\	0	1	12.74	9.36	9.66	11.56	11.56
16	76	134	1	17	0	1	15.84	13.24	1010	13.24	13.24
17	76	134	0	\	0	1	15.84	13.24	1010	13.24	13.24
18	76	223	1	19	0	1	8.66	8.32	1000	8.32	8.32
19	76	223	0	\	0	1	8.66	8.32	1000	8.32	8.32
20	48	178	1	21	0	1	11.74	11.45	11.66	11.45	11.45
21	48	178	0	\	0	1	11.32	11.22	11.35	11.24	11.24
22	48	134	1	23	1	1	5.36	4.98	4.68	1000	1000
23	48	134	0	\	1	1	9.84	7.62	11.62	1000	1000
24	68	201	0	\	1	1	10.56	1000	10	1000	1000
25	48	201	1	26	0	1	11.82	10.55	10.36	10.55	10.55
26	48	201	0	\	0	1	13.03	11.77	10.74	11.77	11.77
27	0	67	1	28	0	1	8.96	8.08	8.29	8.08	8.08
28	0	67	0	\	0	1	10.85	9.41	10.59	9.41	9.41
29	45	245	1	30	0	1	7.68	7.59	7.77	7.12	7.12
30	45	245	0	\	0	1	7.93	7.47	7.2	6.89	6.89
31	0	245	1	32	0	1	1000	16.42	21.83	17.81	17.81
32	0	245	0	\	0	1	1000	16.42	21.83	17.81	17.81
33	0	67	0	\	0	1	9.56	1000	11	1000	1000
34	0	112	1	35	0	1	15.98	15.48	15.21	1000	1000
35	0	112	0	\	0	1	14.87	12.89	16.94	1000	1000
36	22	156	0	\	0	1	11.65	11.45	11.66	11.45	11.45
37	89	201	1	38	0	1	11.39	11.22	11.35	11.24	11.24
38	89	201	0	\	0	1	9.37	1000	1000	8.82	8.82
39	0	178	1	40	1	1	35.17	33.87	36.74	30.92	30.92
40	0	178	0	\	1	1	16.21	11.98	15.75	15.75	15.75
41	82	223	1	42	1	1	45.78	39.55	40.33	37.81	37.81
42	82	223	0	\	1	1	6.74	5.87	6.23	4.85	4.85
43	82	223	1	44	1	1	18.46	18.34	19.29	16.69	16.69
44	82	223	0	\	1	1	6.88	7.02	1000	6.59	6.59
45	82	223	1	46	1	1	27.45	1000	1000	22.92	22.92
46	82	223	0	\	1	1	27.45	1000	1000	22.92	22.92

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

Table 4.7 Ready times and initial RoHS status for assembly lines

	Line 1	Line 2	Line 3	Line 4	Line 5
Ready time	1.52	0	1.78	2.66	0
Initial RoHS status	0	1	0	0	1

It should be noted that in practice, the scheduling problem should be solved each time when there is new production orders in the planning period. All the jobs that have not been processed should be scheduled. However, the jobs that have been started on a particular line and have not been finished are not considered in the scheduling. The ready time of an assembly line is set to be the finish time of the unfinished job on the line. Accordingly, the initial RoHS status of the lines should be the RoHS status of these unfinished jobs.

The proposed genetic algorithm, the GA-N, is used to solve the case problem. The algorithm is run only once and terminates when there is no improvement on the best chromosome during 3000 generations. The detailed schedule obtained by the proposed GA is shown in Figure 4.9.

The schedule obtained by the GA is compared with that obtained by the scheduling staff of the company (as shown in Figure 4.10). The proposed GA solves the problem with 64.2 seconds. Based on the schedule obtained by the proposed GA, all the jobs are finished before their due dates. The makespan obtained by the GA is 131.22 hours, which is 8.88% shorter than that obtained by the company scheduling staff, which is 144 hours.

The results are much encouraging, and the company managers are satisfied. Reduced makespan means higher production efficiency. The production capacity of the company can be increased without further investment on the expensive assembly equipment. In a make-to-order environment, larger production capacity is crucial for PCB manufacturers to gain more profits and become more competitive.

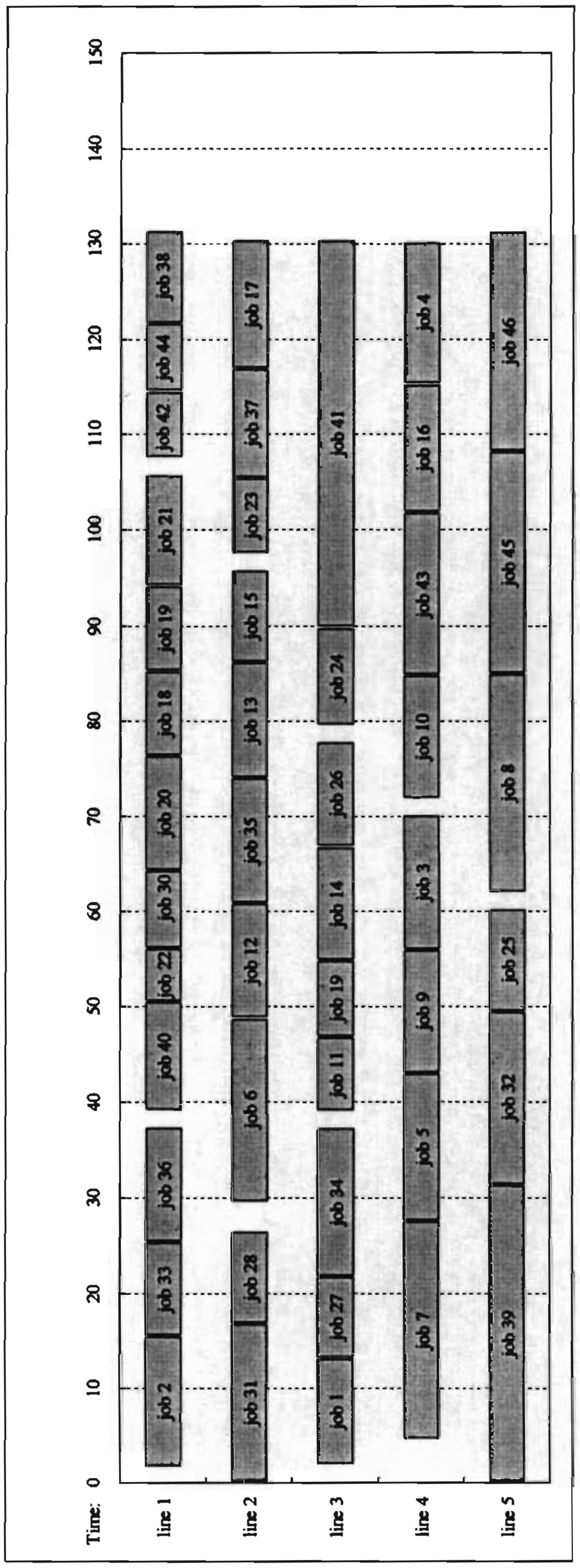


Figure 4.9 The schedule for the case problem obtained by the proposed GA

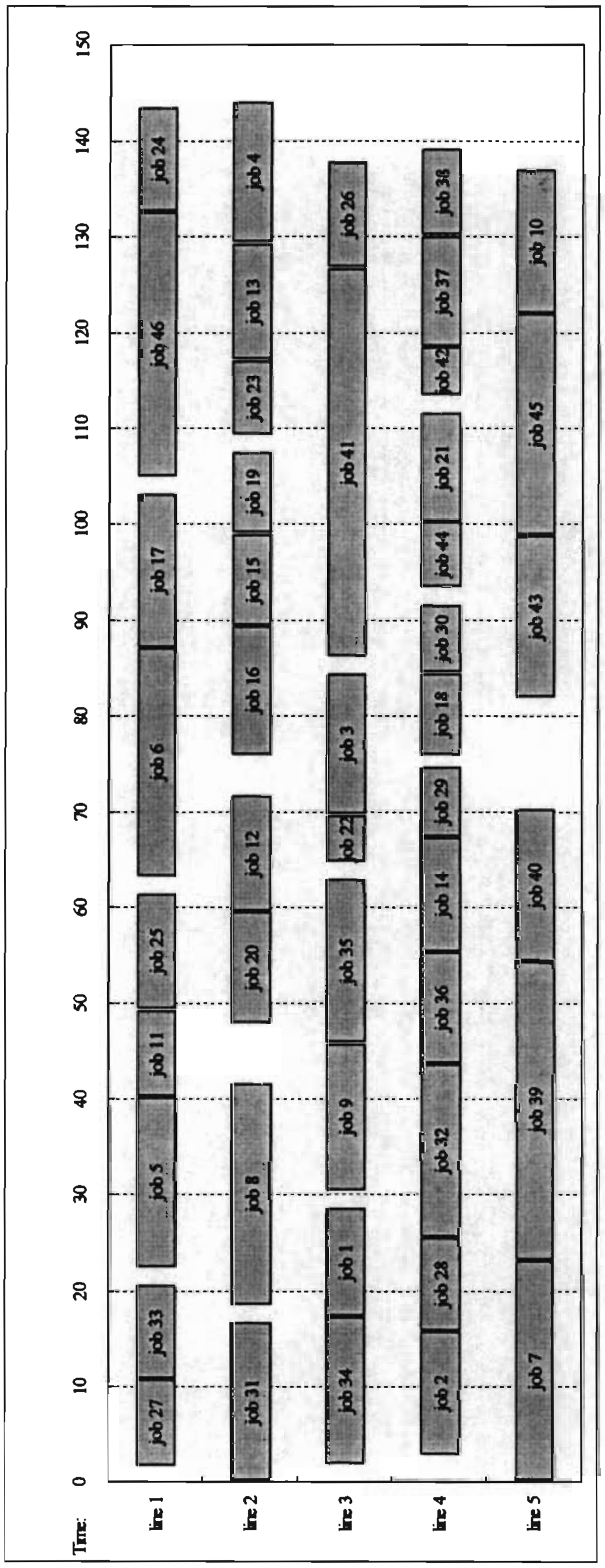


Figure 4.10 The schedule for the case problem obtained by the scheduling staff in the company

4.6 SUMMARY

This chapter investigated the Multi-Line Scheduling Problem (MLSP) in PCB assembly and proposed an effective solution method for the problem. Some main remarks can be summarized as follows.

1. A Mixed Integer Linear Programming (MILP) model has been established for the multi-line scheduling problem in PCB assembly. The objective of the model considers both due date requirement and production efficiency while giving a higher priority to the former. Precedence constraints between jobs and sequence-dependent setup times are also considered to make the model more realistic and applicable.
2. Some test instances have been generated and used to verify the established model. The instances are solved by a commercial LP solver, CPLEX. The results show that the established mathematical model is able to find optimal solution while satisfying all the practical constraints. However, the computational complexity is shown to be extremely great. CPLEX fails to solve the test instance with only 12 jobs and 4 lines within 240 hours. Exact solutions are impossible for real-sized problems.
3. In order to solve the problem efficiently, a GA method has been developed. Because of the great complexity of the problem, a new replacement strategy has been proposed to improve the performance of the GA. Experimental tests show that the algorithm using the new replacement strategy achieves a good balance between exploration and exploitation, and obtains significantly better solutions than the algorithm using the common replace-worst strategy.

4. The proposed GA method obtains the optimal results for all the instances that can be solved by CPLEX. The computational times for the GA method are short. The results show that the proposed GA method can solve the multi-line scheduling problem both effectively and efficiently.
5. A case study on solving a realistic scheduling problem in a PCB company is conducted. The proposed method is able to find an effective scheduling result for the problem. Compared to the solution obtained by the scheduling staff in the company, the proposed method obtains a better schedule with a makespan that is 8.88% shorter, while satisfying the due date requirement. The production capacity of the company can be increased without further investment on the equipment or sacrifice of customer satisfaction.

The solution method proposed in this chapter has shown to be able to solve the multi-line scheduling problem both effectively and efficiently. Since the established model considers realistic constraints and a reasonable objective in PCB assembly environment, the results are highly valuable to PCB manufacturers. The research results in this chapter and those in Chapter 3 are complementary to each other, in that the production efficiency of each line can be improved by solving the component allocation problem, while the production efficiency of the whole assembly shop can be achieved by solving the multi-line scheduling problem. In Chapter 5, this research will be concluded by summarizing the main achievements, academic contributions, and possible benefits to industry. Some possible future work related to this research will also be given.

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

5.1 CONCLUSIONS

In this research, two important planning problems in PCB assembly have been addressed, i.e., the Component Allocation Problem (CAP) and the Multi-Line Scheduling Problem (MLSP). The component allocation problem is vital for improving the throughput of a PCB assembly line, while the multi-line scheduling problem is important for improving overall production efficiency and meeting due date requirement.

The component allocation problem is to allocate components required by a PCB to different machines in an assembly line, so that the line cycle time is minimized. The component allocation problem is found to be interrelated with the machine optimization problems, i.e., the feeder arrangement problem and the placement sequencing problem, for the solutions to the component allocation problem are eventually influenced by the solutions to the machine optimization problems for each machine in the line. In order to reduce the computational complexity caused by this interrelationship, a decomposed solution strategy has been proposed. The effectiveness and efficiency of the proposed solution strategy for the component allocation problem has been examined in the experimental tests.

The multi-line scheduling problem is to schedule PCB batches on multiple assembly lines, so that the sum of weighted tardiness and weighted makespan is minimized. The specific multi-line scheduling problem is shown to be a special type of unrelated Parallel-Machine Scheduling Problem (PMSP) with sequence-dependent setup times and precedence constraints, which is *NP*-hard. Due to the complexity of the problem, a specific Genetic Algorithm (GA) has been proposed to solve the problem. Experimental tests have been conducted to examine the effectiveness and efficiency of the GA method.

The main conclusions of this research are as follows.

1. The Component Allocation Problem (CAP) is extremely complicated for the evaluation of the solutions involves solving the machine optimization problems for each machine in the line. The proposed solution strategy aims to solve the component allocation problem without tackling the machine optimization problems. The solution strategy relies on a regression-based placement time estimator, which can estimate the placement time for each machine in the line without solving the machine optimization problems. Based on this placement time estimator, algorithms or heuristics can be devised to solve the component allocation problem without tackling the machine optimization problems.
2. A placement time estimator for a turret-type placement machine has been established to examine the feasibility of the solution strategy for the component allocation problem. The estimator is based on a linear regression model, which considers all the influential factors that may affect the placement time, including the number of components, the number of component types, and the closeness of the components. The regression model

is calibrated using a data set which is obtained through experiments. Statistical analysis shows that the placement time estimator has a high R^2 value of 0.99, showing that the estimator can yield accurate estimates of placement time, without solving the machine optimization problems.

3. The proposed solution strategy for the component allocation problem is implemented through the development of a specific genetic algorithm, which uses the established placement time estimator for solution evaluation. Based on the placement time estimator, the genetic algorithm is able to consider all the influential factors that affect the placement time for each machine when solving the component allocation problem. Experiments on solving some problem instances show that the proposed solution method can achieve better solutions than those obtained by the software provided by the machine vendor. The results are encouraging, especially because the GA obtains the solutions without calculating the exact placement times through simulation, as the vendor software does. Even better solutions could be expected if the GA solutions are further improved by some adjustments based on the simulated placement times.
4. A Mixed Integer Linear Programming (MILP) model has been established for the multi-line scheduling problem in PCB assembly. A reasonable objective and some practical constraints are considered in the model. The objective considers both due date satisfaction and production efficiency while giving a higher priority to due date satisfaction. Precedence constraints between jobs and sequence-dependent setup times are also considered to make the model

more realistic and applicable. Some test instances are generated and used to verify the established model. The instances are solved by a commercial LP solver, CPLEX. The testing results show that the established model is able to find effective solutions to the scheduling problem. However, while some small instances could be solved to optimality, the computational complexity is shown to be extremely great. Exact solutions are impossible for real-sized problems.

5. In order to solve the multi-line scheduling problem efficiently, a specific genetic algorithm has been developed. In order to improve the performance of the GA, a new replacement strategy is proposed and used in the genetic algorithm. Experimental tests show that the algorithm using the new replacement strategy can obtain much better solutions than the algorithm using the replace-worst replacement strategy, which is one of the most commonly-used replacement strategies. The proposed GA method obtains the optimal results for all the instances that can be solved by CPLEX, with much less computational time. A case study on solving a realistic scheduling problem is also conducted. The results show that the proposed GA method is able to solve realistic multi-line scheduling problems. Compared to the solution obtained by the scheduling staff in the investigated manufacturer, the proposed GA method obtains a schedule with much smaller makespan while ensuring due date satisfaction. This shows that the proposed solution method can help the manufacturers to improve both production efficiency and customer satisfaction.

5.2 ACADEMIC CONTRIBUTIONS

The planning problems in PCB assembly have been investigated by numerous researchers. Relative papers on this topic have been published in the operations research, industrial engineering and production management literature. The academic contributions of this research can be summarized as follows.

First of all, the decomposed solution strategy proposed for the component allocation problem is an attempt to solve the higher-level planning problems effectively by relaxing the dependency on the lower-level planning problems. The successful implementation of this solution strategy indicates its potential use for solving other hierarchical planning problems.

Second, a placement time estimator that can estimate the placement time for a PCB placement machine rapidly and accurately has been successfully established. Although the placement time is determined by the solutions to the complicated machine optimization problems, i.e., the feeder arrangement and placement sequence, it can be accurately estimated using the regression method. The successful development of the estimator provides an efficient way for obtaining objective values for a complicated planning problem without solving it. The approach might be important for applications like decision support systems.

Third, results of experimental tests indicate that, for the genetic algorithm for the component allocation problem, the swap mutation operator and the fitness function considering both cycle time and total line time perform significantly better than the random-point mutation operator and the fitness function considering only cycle time, respectively. These findings are thought to be due to the line balancing property of the component allocation problem. Therefore, the results may be transferable to other types of line balancing problems, although a further research is required.

Fourth, a complete mathematical model has been established for the multi-line scheduling problem in PCB assembly, which fills up a gap in the relative research. The established model is different from existing models in the literature, in that it considers a more reasonable objective and some practical constraints, which are realistic in many PCB manufacturing environments.

Finally, a new replacement strategy has been proposed for the genetic algorithm and shown to be effective for improving the algorithm performance in solving the multi-line scheduling problem. The new replacement strategy can achieve a good balance between solution exploration and exploitation and thus improve the performance of the algorithm. The new replacement strategy may be helpful for solving other complicated optimization problems.

5.3 POSSIBLE BENEFITS TO INDUSTRY

The planning problems investigated in this research are quite common in many PCB manufacturers. The established approaches are important for the manufacturers to improve production efficiency while maintaining customer satisfaction.

The established approach for the multi-line scheduling problem helps the manufacturers to make an effective schedule that considers both due date satisfaction and makespan reduction. Some realistic considerations are made in the established model so that the solution is more applicable. These considerations include the sequence-dependent setup times, precedence requirements between jobs, ready times and due dates for the jobs, ready times for the assembly lines, etc. In addition, the approach allows PCB manufacturers to define relative importance for each order while optimizing the schedule. A realistic problem case is studied and it shows that the established approach can achieve a better scheduling result than the scheduling staff in the investigated manufacturing company. The makespan is reduced by 8.88%, indicating that production efficiency is improved. The production capacity of the company can be increased without further investment on the expensive assembly equipment.

The established approach for the component allocation problem is important for improving the line throughput when a job is assigned to a specific assembly line. Experimental tests show that the proposed method can achieve better solutions than those obtained by the existing software provided by machine vendor. Since the established approach for the component allocation problem relies on the regression-based placement time estimator, knowledge on technological specifications of the placement machines is not necessary. Nevertheless, even better solutions could be expected if the technological specifications about the machine are known. Furthermore, the proposed method is applicable to assembly lines with mixed-vendor machines, to which the vendor software is not applicable.

5.4 FUTURE WORK

Due to the large variety of manufacturing environments in PCB assembly, certain limitations exist in this research. Some possible future work related to this research is suggested as follows.

1. The placement time estimators for other types of placement machines should be established.

   In this research, the placement time estimator for a turret-type placement machine is established and shown to be effective for obtaining accurate estimates of placement time. However, due to different operation modes, the specification of the estimator for a different machine, e.g., a sequential pick-and-place machine, may be different. The effectiveness of the regression method should be examined for other machine types.

2. Other approaches can be tried for the placement time estimators.

   Although the regression method is shown to be effective for estimating the placement time for the turret-type machine, the effectiveness may not hold. Other methods can be considered in the development of the estimators in the future. One alternative is the neural network approach, which has shown to be as effective as the linear regression approach for developing TSP minimum tour estimator [Kwo95].

3. The component allocation problem with feeder duplication should be investigated.

   The component allocation problem investigated in this research does not consider feeder duplication, which allows the same components to be assigned to more than one machine in the line. By intuition, feeder duplication is beneficial for reducing the line cycle time, although some PCB manufacturers do not consider feeder duplication due to the expensiveness of the feeders. However, the complexity of the component allocation problem greatly increases with feeder duplication for the problem not only determines the assignment of feeders to machines, but also the allocation of the same components to different machines.

4. Integration of the approaches established in this research with the existing machine vendor software should be done to facilitate practical use in the industry.

   The solutions to the multi-line scheduling problem and the component allocation problem serve as input to the lowest-level machine optimization problems, i.e., the feeder arrangement and placement sequence for each single machine. Most PCB manufacturers currently use software packages provided by machine vendors for solving the machine optimization problems. Therefore, integration of the established approaches with the existing machine optimization software is necessary for practical use in the industry.

Due to the ever-advancing technologies in PCB assembly, planning problems in PCB assembly are becoming more and more complicated. Solutions to these problems have been and will always be essential for improving production efficiency and maintaining customer satisfaction. The approaches and methodologies developed in this research are the results of a successful application of operations research techniques in PCB assembly planning. Both industrial benefits and academic contributions of this research have been illustrated. Nevertheless, much research work should be done in the future to adopt the developed approaches in various and variable planning situations.

REFERENCES

[Aar89] Aarts, E. H. L., 1989, *Simulated Annealing and Boltzmann Machine: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley.

[Aar97] Aarts, E. and Lenstra, J. K., 1997, *Local Search in Combinatorial Optimization*, Wiley.

[Abd88] Abdul-Razaq, T. S. and Potts, C. N., 1988, Dynamic programming state–space relaxation for single–machine scheduling, *Journal of the Operational Research Society*, 39(2), 141–152.

[Ahm99] Ahmadi, R. H. and Mamer, J. W., 1999, Routing heuristics for automated pick and place machines, *European Journal of Operational Research*, 117, 533–552.

[Alt00] Altinkemer, K., Kazaz, B., Köksalan, M. and Moskowitz, H., 2000, Optimization of printed circuit board manufacturing: Integrated modeling and algorithms, *European Journal of Operational Research*, 124, 409–421.

[Amm97] Ammons, J. C., Carlyle, M., Cranmer, L., DePuy, G., Ellie, K., McGinnis, L.F., Tovey, C. A. and Xu, H., 1997, Component allocation to balance workload in printed circuit card assembly systems, *IIE Transactions*, 29, 265–275.

[Ang07] Anghinolfi, D. and Paolucci, M., 2007, Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach, *Computers & Operations Research*, 34(11), 3471–3490.

[Arm07] Armentano, V. A. and de França Filho, M. F., 2007, Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach. *European Journal of Operational Research*, 183, 100–114.

[Ash07] Ashayeri, J. and Selen, W., 2007, A planning and scheduling model for onsertion in printed circuit board assembly, *European Journal of Operational Research*, 183, 909–925.

[Ayo05] Ayob, M., Kendall, G., 2005a. A triple objective function with a chebychev dynamic point specification approach to optimize the SMD placement machine, *European Journal of Operational Research*, 164, 609–626.

[Ayo08] Ayob, M. and Kendall, G., 2008, A survey of surface mount device placement machine optimization: Machine classification, *European Journal of Operational Research*, 186, 893–914.

[Bal88] Ball, M.O. and Magazine, M. J., 1988, Sequencing of insertions in printed circuit board assembly, *Operations Research*, 36, 192–201.

[Bal99] Balakrishnan, A. and Vanderbeck, F., 1999, A tactical planning model for mixed–model electronics assembly operations, *Operations Research*, 47, 3, 395–409.

[Bar77] Barnes, J. W. and Brennan, J. J., 1977, An improved algorithm for scheduling jobs on identical machines, *IIE Transactions*, 9, 25–31.

[Bar06] Bard, J. F. and Rojanasoonthon, S., 2006, A branch-and-price algorithm for parallel machine scheduling with time windows and job priorities, *Naval Research Logistics*, 53, 1, 24–44.

[Bea94] Bean, J., 1994, Genetic algorithms and random keys for sequencing and optimization, *ORSA Journal on Computing*, 6, 154–160.

[Bel94] Belouadah, H. and Potts, C. N., 1994, Scheduling identical parallel machines to minimize total weighted completion time, *Discrete Applied Mathematics*, 48, 201–218.

[Ben90] Ben–Arieh, D. and Dror, M., 1990, Part assignment to electronic insertion machines: Two machine case, *International J ournal of Production Research*, 28(7), 1317–1327.

[Blu01] Blum, C. and Roli, A., 2001, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, Technical Report TR/IRIDIA/2001-13, IRIDIA, Belgium.

[Bro96] Broad, K., Mason, A., Rönnqvist, M. and Frater, M., 1996, Optimal robotic component placement, *Journal of t he O perational R esearch Society*, 47, 1343–1354.

[Çat06] Çatay, B., Vakharia, A. J., Erengüç, S. S., 2006, Printed circuit board scheduling in an openshop manufacturing environment, *International Journal of Advanced Manufacturing Technology*, 29, 980–989.

[Che90] Cheng, T. C. E. and Sin, C. C. S., 1990, A state-of-the-art review of parallel–machine scheduling research, *European J ournal of Operational Research*, 47 271–292.

[Chi93] Chien, T. W., 1992, Heuristic procedures for practical–sized uncapacitated location–capacitated routing problems, *Decision Sciences*, 24, 995–1021.

[Coo71] Cook, S. A., 1971, The complexity of theorem–proving procedures, *Proceedings of the 3rd annual ACM Symposium on Theory of Computing*, 151–158.

[Cra90] Crama, Y., Kolen, A. W. J., Oerlemans, A. G. and Spieksma, F. C. R., 1990, Throughput rate optimization in the automated assembly of printed circuit boards, *Annals of Operations Research*, 26, 455–480.

[Cra97] Crama, Y., Flippo, O. E., Klundert, J. V. D. and Spieksma, F. C. R., 1997, The assembly of printed circuit boards: A case with multiple machines and multiple board types, *European Journal of Operational Research*, 98, 457–472.

[Cra02] Crama, Y., Klundert, J. V. D. and Spieksma, F. C. R., 2002, Production planning problems in printed circuit board assembly, *Discrete Applied Mathematics*, 123, 339–361.

[Csa00a] Csaszar, P., Tirpak, T. M. and Nelson, P. C., 2000a, Optimization of a high–speed placement machine using tabu search algorithms, *Annals of Operations Research*, 96, 125–147.

[Csa00b] Csaszar, P., Nelson, P. C., Rajbhandari, R. R. and Tirpak, T. M., 2000b, Optimization of automated high–speed modular placement machines using knowledge-based systems, *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 30, 4, 408–417.

[Dav91] Davis, L., 1991, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

[Del95] Dell'Amico, M. and Martello, S., 1995, Optimal scheduling of tasks on identical parallel processors, *ORSA Journal on Computing*, 2(7), 191–200.

[Dej06] De Jong, K. A., 2006, *Evolutionary Computation: A Unified Approach*, The MIT Press, London.

[Dep01] DePuy, G. W., Savelsbergh, M. W. P., Ammons, J. C. and McGinnis, L. F., 2001, An integer programming heuristic for component allocation in printed circuit card assembly systems, *Journal of Heuristics*, 7, 351–369.

[Des94] De Souza, R. and Lijun, W., 1994. CPS: A productivity tool for component placement in multi–head concurrent operation PCBA machines, *Journal of Electronics Manufacturing*, 4 (2), 71–79.

[Deo02] Deo, S., Javadpour, R. and Knapp, G. M., 2002, Multiple setup PCB assembly planning using genetic algorithms, *Computers & Industrial Engineering*, 42, 1–16.

[Dre84] Drezner, Z. and Nof, S., 1984, On optimizing bin picking and insertion plans for assembly robots, *IIE Transactions*, 16, 262–270.

[Dum05] Duman, E., 2005, A note on 'balancing printed circuit board assembly line systems', *International Journal of Production Research*, 43(18), 3955–3957.

[Egb96] Egbelu, P. J., Wu, C. T. and Pilgaonkar, R., 1996, Robotic assembly of printed circuit boards with component feeder location consideration. *Production Planning & Control*, 7, 162–175.

[Ell01] Ellis, K. P., Vittes, F. J. and Kobza, J. E., 2001, Optimizing the performance of a surface mount placement machine, *IEEE Transactions on Electronics Packaging Manufacturing*, 24, 3, 160–170.

[Ell02] Ellis, K. P. and Bhoja, S., 2002, Optimization of the assignment of circuit cards to assembly lines in electronics assembly, *International Journal of Production Research*, 40, 11, 2690–2631.

[Elm74] Elmaghraby, S. E. and Park, S. H., 1974, Scheduling jobs on a number of identical machines, *IIE Transactions*, 6, 1–13.

[Feo95] Feo, T. A. and Bard, J. F., 1995, Facility-wide planning and scheduling of printed wiring board assembly, *Operations Research*, 43(2), 219–230.

[Fou93] Foulds, L. R. and Hamacher, H. W., 1993, Optimal bin location and sequencing in printed circuit board assembly. *European Journal of Operational Research*, 66, 279–290.

[Fra94] França, P. M., Gendrau, M., Laporte, G. and Muller, F. M., 1994, A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective, *Computers & Operations Research*, 21(2), 205–210,.

[Fra04] Frangioni, A., Necciari, E. and Scutella, M. G., A multi–exchange neighborhood for minimum makespan parallel machine scheduling problems, *Journal of Combinatorial Optimization*, 8(2), 195–220.

[Gla94] Glass, C. A., Potts, N. and Shade, P., 1994, Unrelated parallel machine scheduling using local search, *Mathematical and Computational Modeling*, 20, 41–52.

[Glo89] Glover, F., 1989, Tabu search: Part I, *ORSA Journal on Computing*, 1, 190–206.

[Glo90] Glover, F., 1990, Tabu search: Part II, *ORSA Journal on Computing*, 2, 4–32.

[Glo93] Glover, F., Taillard, E. and De Werra, D., 1993), A user's guide to tabu search, *Annals of Operations Research*, 41(3), 3–28.

[Gol89] Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley.

[Gra79] Graham, R. L., Lawler, E. L., Lenstra, J. K. and Rinnooy Kan A. H. G., 1979, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5, 287–326.

[Gro00] Gronalt, M. and Zeller, R., 2000, Component allocation and job sequencing for two series–connected SMD placement machines, *International Journal of Production Research*, 38(2), 409–427.

[Gru03] Grunow, M., Günther, H. O. and Schleusener, M., 2003, Component allocation for printed circuit board assembly using modular placement machines, *International Journal of Production Research*, 41(6), 1311 –1331

[Gru04] Grunow, M., Günther, H. O., Schleusener, M. and Yilmaz, I. O., 2004, Operations planning for collect-and-place machines in PCB assembly, *Computers & Industrial Engineering*, 47, 409–429.

[How03] Ho, W. and P. Ji., 2003, Component scheduling for chip shooter machines: A hybrid genetic algorithm approach. *Computers and Operations Research*, 30, 2175–2189.

[How04] Ho, W. and P. Ji., 2004, A hybrid genetic algorithm for component sequencing and feeder arrangement, *Journal of I ntelligent Manufacturing*, 15, 307–315.

[How07] Ho, W., Ji, P. and Wu, Y., 2007, A heuristic approach for component scheduling on a high–speed PCB assembly machine, *Production Planning & Control*, 18(8), 655 – 665

[Hüb94] Hübscher, R. and Glove, F., Applying tabu search with influential diversification to multiprocessor scheduling, *Computers and Operations Research*, 8(21), 877–884.

[Jip01] Ji, P., Sze, M. T., Lee, W.B., 2001, A genetic algorithm of determining cycle time for printed circuit board assembly lines, *European Journal of Operational Research*, 128, 175–184.

[Jip05] Ji, P. and Ho, W., 2005, PCB assembly line assignment: a genetic algorithm approach, *Journal of M anufacturing T echnology Management*, 16(6), 682–692.

[Joh90] Johnson, D. S., 1990, Local optimization and the traveling salesman problem, In: Goos, G. and Hartmanis, J.: *Automata, L anguages and*

Programming, Springer, Heidelberg.

[Kar72] Karp, R. M., 1972, Reducibility among combinatorial problems, In Miller, R. E. and Tatcher, J. W., eds.: *Complexity of C omputer Computations*, Plenum Press, New York, 85–103.

[Kho00] Khoo, L. P. and Loh, K. M., 2000, A genetic algorithms enhanced planning system for surface mount PCB assembly, *International Journal of Advanced Manufacturing Technology*, 16, 289–296.

[Kim04] Kim, K. M. and Park, T. H., 2004, PCB assembly optimization of chip mounters for multiple feeder assignment, *SICE Annual Conference in Sapporo*, 1425–1430.

[Kir83] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., 1983, Optimization by simulated annealing, *Science*, 220(4598), 671–680.

[Klo00] Klomp, C., Klundert, J. V. D., Spieksma, F. C. R. and Voogt, S., 2000, The feeder rack assignment problem in PCB assembly: A case study. *International Journal of Production Economics*, 64, 399–407.

[Kod04] Kodek, D. M. and Krisper, M., 2004, Optimal algorithm for minimizing production cycle time of a printed circuit board assembly line, *International J ournal of P roduction R esearch*, 42(23), 5031–5048.

[Kul07] Kulak, O., Yilmaz, I. O. and Günther, H. O., 2007, PCB assembly scheduling for collect-and-place machines using genetic algorithms, *International Journal of Production Research*, 45(17), 3949 –3969.

[Kum95] Kumar, R. and Li, H., 1995, Integer programming approach to printed circuit board assembly time optimization, *IEEE Transactions on Components, Packaging, and Manufacturing Technology-Part B*, 18 (4), 720–727.

[Kwo95] Kwon, O., Golden, B., and Wasil, E., 1995, Estimating the length of the optimal TSP tour: An empirical study using regression and neural networks, *Computers and Operations Research*, 22, 1039–1046.

[Len80] Lenstra, J. K. and Rinnooy Kan, A. H. G., 1980, *An Introduction to Multi-processor Scheduling*, Matematisch Centrum, Amsterdam.

[Lap88] Laporte, G., 1988, Location–routing problems, In: Golden, B. and Assad, A., *Vehicle Routing: Methods and Studies*, Elsevier Science Publishers, Amsterdam.

[Lap00] Lapierre, S. D., Debargis, L. and Soumis, F., 2000, Balancing printed circuit board assembly line systems, *International Journal of Production Research*, 38 (16), 3899–3911.

[Law69] Lawler, E. L. and Moore, J. M., 1969, A functional equation and its application to resource allocation and sequencing problems, *Management Science*, 16, 77–84.

[Lei89] Leipälä, T. and Nevalainen, O., 1989, Optimization of the movements of a component placement machine, *European Journal of Operational Research*, 38, 167–177.

[Leu82] Leung, J. Y. T., 1982, On scheduling independent tasks with restricted execution times, *Operations Research*, 30, 163–171.

[Leu93] Leu, M. C., Wong, H. and Ji, Z., 1993, Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm, *Journal of Electronic Packaging*, 115, 424–432.

[Lis07] Li, S., Hu, C. and Tian, F., 2007, Enhancing optima feeder assignment of the multi-head surface mounting machine using genetic algorithms, *Applied Soft Computing*, 8, 522–529.

[Mag99] Magyar, G., Johnsson, M. and Nevalainen, O., 1999, On solving single machine optimization problems in electronics assembly, *Journal of Electronics Manufacturing*, 9(4), 249–267.

[Man99] Man, K. F., Tang, K. S. and Kwong, S., 1999, *Genetic A lgorithm: Concepts and Design*, Springer.

[Mar97] Martello, S., Soumis, F. and Toth, P., 1997, Exact and approximation algorithms for makespan minimization on unrelated parallel machines, *Discrete Applied Mathematics*, 2(87), 169–188.

[Mcg92] McGinnis, L. F., Ammons, J. C., Carlyle, M., Cranmer, L., DePuy, G. W., Ellis, K. P., Tovey, C. A. and Xu, H., 1992, Automated process planning for printed circuit card assembly, *IIE Transaction*, 24, 18–30.

[Mok01] Mokotoff, E., 2001, Parallel machine scheduling problems: a survey, *Asia-Pacific Journal of Operational Research*, 18, 193–242.

[Moy96a] Moyer, L. K., Gupta, S. M., 1996a, SMT feeder slot assignment for predetermined component placement paths, *Journal of E lectronics Manufacturing,* 6, 173–192.

[Moy96b] Moyer, L. K. and Gupta, S. M., 1996b, Simultaneous component sequencing and feeder assignment for high speed chip shooter machines, *Journal of Electronics Manufacturing*, 6, 271–305.

[Moy97] Moyer, L. K. and Gupta, S. M., 1997, An efficient assembly sequencing heuristic for printed circuit board configurations, *Journal of Electronics Manufacturing*, 72, 143–160.

[Mok01] Mokotoff, E., 2001, Parallel machine scheduling problems: a survey, *Asia-Pacific Journal of Operational Research*, 18, 193–242.

[Nes08] Nessah, R., Yalaoui, F. and Chu, C. B., 2008, A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates, *Computers & Operations Research*, 35, 1176–1190.

[Ong99] Ong, N. S. and Khoo, L. P., 1999, Genetic algorithm approach in PCB assembly, *Integrated Manufacturing Systems*, 10, 256–265.

[Pha00] Pham, D. T., and Karaboga, D., 2000, *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer, New York.

[Pie96] Piersma, N. and van Dijk, W., 1996, A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search, *Mathematical and Computational Modeling*, 24, 11–19.

[Ree95] Reeve, C. R. and Beasley, J. E., 1995, Chapter 1. Introduction, In: Reeves, C. R., *Modern Heuristic Techniques for Combinatorial Problems*, McGraw-Hill.

[Raw98] Rawlings, J. O., Pantula, S. G. and Dickey, D. A., *Applied Regression Analysis: A Research Tool, Second Edition*, Springer, New York.

[Roj05] Rojanasoonthon, S. and Bard, J., A GRASP for parallel machine scheduling with time windows, *Informs Journal of Computing*, 17(1), 32–51.

[Rot66] Rothkoph, M. H.,1966, Scheduling independent tasks on parallel processors, *Management Science*, 5(12), 437–447.

[Sar88] Sarin, S. C., Ahn, S. and Bishop, A. B., 1988, An improved branching scheme for the branch and bound procedure of scheduling *n* jobs on *m* parallel machines to minimized total weighted flowtime, *International Journal of Production Research*, 26, 1183–1191.

[Scu00] Scutella, M. G., Frangioni, A. and Necciari, E., 2000, Multi-exchange algorithms for the minimum makespan machine scheduling problem, *17th European Conference on Operational Research*, Budapest.

[Shi08] Shim, S. O. and Kim, Y. D., 2008, A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property, *Computers & Operations Research*, 35, 863–875.

[Sil80] Silver, E., Vidal, R. V. and de Werra, D., 1980, A tutorial on heuristic methods, *European Journal of Operational Research*, 5, 153–162.

[Siv99] Sivrikaya–Serifoglu, F. and Ulusoy, G., 1999, Parallel machine scheduling with earliness and tardiness penalties, *Computers and Operations Research*, 26 (8), 773–787.

[Smu98] Smutnicki, C., 1998, Various optimizers for single–stage production, *Naval Research Logistics Quarterly*, 3, 59–66.

[Soh96] Sohn, J. and Park, S., 1996, Efficient Operation of a Surface

Mounting Machine with a Multihead Turret, *International Journal of Production Research*, 34(4), 1131–1143.

[Sot95] Sotskov, Y. N. and Shaklevich, N. V., 1995, NP–hardness of shop scheduling problems with three jobs, *Discrete Applied Mathematics*, 59, 237 –266.

[Sou99] Sourd, F., 1999, *Scheduling Tasks on Unrelated Machines: Large Neighborhood Improvement Procedures, Manuscript LIP6*, Université Pierreet Marie Curie, Paris.

[Sri98] Srivastava, B., 1998, An effective heuristic for minimizing makespan on unrelated parallel machines, *Journal of the Operational Research Society*, 49(8), 886–894.

[Sun05] Sun, D. S., Lee, T. E and Kim, K. H., 2005, Component allocation and feeder arrangement for a dual-gantry multi-head surface mounting placement tool, *International Journal of Production Economics*, 95, 245–264.

[Sur96] Suresh, V and Chaudhuri, D., 1996, Bicriteria scheduling problem for unrelated parallel machines, *Computers and Industrial Engineering*, 30 (1), 77–82.

[Tir00] Tirpak, T. M., Nelson, P. C. and Aswani, A. J., 2000, Optimization of revolver head SMT machines using adaptive simulated annealing (ASA), *Electronics Manufacturing Technology Symposium, Twenty-Sixth IEEE/CPMT*, 214–220.

[Van93] Van de Velde, S. L., 1993, Duality–based algorithms for scheduling unrelated parallel machines, *ORSA Journal on Computing*, 5, 192–205.

[Wan99] Wang, W., Nelson, P.C. and Tirpak, T.M., 1999, Optimization of high–speed multistation SMT placement machines using evolutionary algorithms., *IEEE Transactions on Electronics Packaging Manufacturing*, 22 (2), 137–146.

[Yal02] Yalaoui, F. and Chu, C. B., 2002, Parallel machine scheduling to minimize total tardiness, *International Journal of Production Economics*, 76(3), 265–279.

[Yeo96] Yeo, S. H., Low, C. W. and Yong, K. H., 1996, A rule–based frame system for concurrent assembly machines, *International Journal of Advanced Manufacturing Technology*, 12, 370–376.

[Zan89] Zanakis, S. H., Evans, J. R. and Vazacopoulos, A. A., 1989, Heuristic methods and applications: A categorized survey, *European Journal of Operational Research*, 43, 88–110.

[Zho07] Zhou, H., Li, Z. D. and Wu, X. J., 2007, Scheduling unrelated parallel machine to minimize total weighted tardiness using ant colony optimization, *IEEE International Conference on Automation and Logistics*, 132–136.

APPENDIX I

DATA FOR THE TEST PCB WITH 61 COMPONENTS

Number of components: 61

Number of component types: 7

Table I-1 Data for the PCB with 61 components

Component Number	Type	*X*	*Y*	Component Number	Type	*X*	*Y*
1	1	303	167	32	3	102	319
2	5	348	310	33	3	278	198
3	4	151	297	34	3	352	293
4	5	81	327	35	3	395	253
5	6	106	109	36	5	344	156
6	2	91	347	37	4	340	298
7	2	92	90	38	5	22	124
8	1	300	271	39	2	210	83
9	5	445	118	40	2	156	43
10	1	121	196	41	3	286	208
11	2	277	341	42	2	394	73
12	5	179	346	43	2	247	353
13	2	443	121	44	3	121	3
14	1	381	90	45	4	171	325
15	3	163	45	46	3	363	331
16	1	29	211	47	2	239	132
17	5	164	342	48	5	91	212
18	4	225	39	49	5	180	44
19	2	113	294	50	2	90	308
20	7	293	244	51	3	61	280
21	2	73	13	52	4	243	353
22	5	245	212	53	4	151	352
23	4	95	112	54	3	255	259
24	5	385	148	55	1	434	179
25	5	1	251	56	1	397	341
26	1	242	233	57	4	274	225
27	1	299	18	58	2	1	259
28	3	167	209	59	2	105	272
29	4	269	148	60	4	81	300
30	2	381	117	61	1	269	74
31	2	198	260				

APPENDIX II

THE MLSP INSTANCE Test-n10k4 AND ITS SOLUTION

Table II-1 Data for the MLSP instance Test-n10k4

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i[f]	process time[g]			
							line 1	line 2	line 3	line 4
1	0	18	0	\	0	2	1000	4.23	5.68	5.24
2	0	11	0	\	1	3	2.45	2.36	2.45	2.65
3	5	15	1	4	0	1	3.79	3.54	3.57	3.97
4	0	9	0	\	0	1	8.96	8.29	1000	8.08
5	5	25	1	6	1	2	10.85	10.59	9.41	9.41
6	0	19	0	\	1	2	7.68	7.77	7.12	7.12
7	0	8	1	8	0	1	1000	1000	8.28	6.89
8	8	25	0	\	1	1	6.42	11.83	7.22	7.90
9	8	40	1	10	0	1	6.21	9.53	8.63	8.01
10	10	27	0	\	1	1	1000	11	10.65	10.41
						ready time of line	0	4	0	2.5
						initial RoHS status of lines[h]	0	1	1	1

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

Optimal solution to Test-n10k4 obtained by CPLEX:

Integer optimal

Objective = 2.23110000000e-001

Solution time = 1982 sec.

Iterations = 24566571

Nodes = 1054239

Variable Name	Solution Value	Variable Name	Solution Value
Cmax	23.11		
x000201	1.000000	x020601	1.000000
x060801	1.000000	x081101	1.000000
x000102	1.000000	x010302	1.000000
x030902	1.000000	x091102	1.000000
x000503	1.000000	x051003	1.000000
x101103	1.000000	x000704	1.000000
x070404	1.000000	x041104	1.000000
w0302	4.77	w0201	1.75
w0601	4.78	w0902	8.58
w1003	12.46	w0404	7.43
w0704	0.27	w0102	0.27
w0503	2.78	w0801	12.73

All other variables are zero.

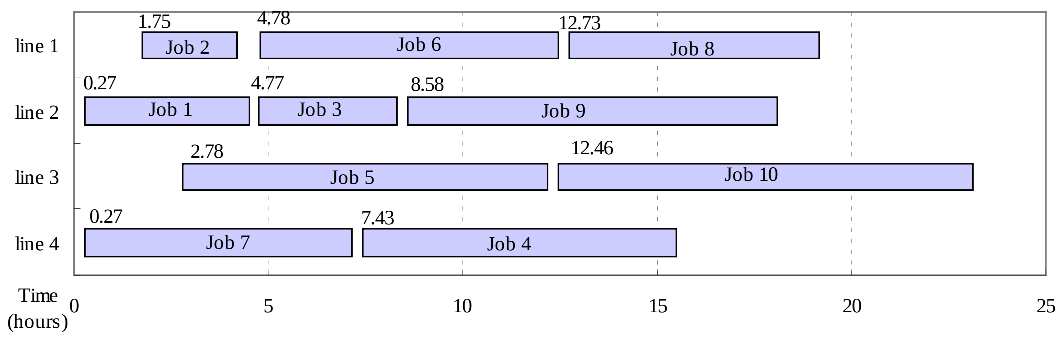


Figure II-1 Optimal schedule for the instance Test-n10k4 shown as Gantt chart

APPENDIX III

THE MLSP INSTANCE Test-n11k3 AND ITS SOLUTION

Table III-1 Data for the MLSP instance Test-n11k3

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i[f]	process time[g]		
							line 1	line 2	line 3
1	5	20	0	\	0	1	6.86	6.21	6.27
2	0	09	0	\	1	1	5.65	5.02	4.89
3	4	15	1	4	0	2	1000	8.75	8.10
4	0	18	0	\	0	2	6.74	6.08	5.75
5	0	25	1	6	1	1	8.41	8.02	8.56
6	0	19	0	\	1	3	5.12	4.87	4.90
7	8	40	0	\	0	2	3.65	3.05	1000
8	10	27	0	\	1	3	6.87	6.12	6.78
9	5	16	0	\	1	1	1000	4.57	3.54
10	0	8	1	11	0	1	5.19	4.54	4.91
11	8	25	0	\	1	1	9.75	9.87	9.51
						ready time of line	1.48	0	0.78
						initial RoHS status of lines[h]	0	1	1

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

Optimal solution to Test-n11k3 obtained by CPLEX:

Integer optimal

Objective = 2.10050000000e-000

Solution time = 9670 sec.

Iterations = 133680879

Nodes = 6875180

Variable Name	Solution Value	Variable Name	Solution Value
L11	1.050000	L01	0.790000
Cmax	26.05		
x001001	1.000000	x100401	1.000000
x041101	1.000000	x111201	1.000000
x000202	1.000000	x020302	1.000000
x030102	1.000000	x010702	1.000000
x071202	1.000000	x000503	1.000000
x050903	1.000000	x090603	1.000000
x060803	1.000000	x081203	1.000000
w0903	9.88	w1001	1.75
w0702	21.06	w0102	14.58
w0202	0.27	w0603	13.69
w0803	18.86	w1101	16.30
w0302	5.56	w0401	7.56
w0503	1.05		

All other variables are zero.

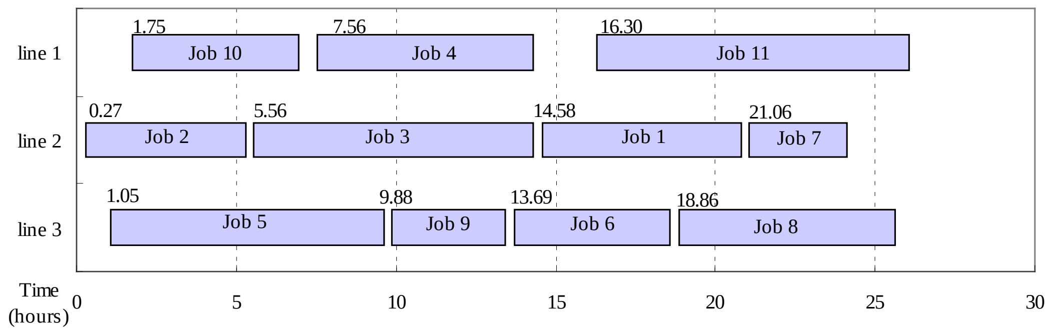


Figure III-1 A Gantt chart for the optimal schedule of Test-n11k3

APPENDIX IV
THE MLSP INSTANCE Test-n11k4 AND ITS SOLUTION

Table IV-1 Data for the MLSP instance Test-n11k4

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i[f]	process time[g] line 1	line 2	line 3	line 4
1	0	18	0	\	0	2	1000	4.23	5.68	5.24
2	0	11	0	\	1	3	2.45	2.36	2.45	2.65
3	5	15	1	4	0	1	3.79	3.54	3.57	3.97
4	0	9	0	\	0	1	8.96	8.29	1000	8.08
5	5	25	1	6	1	2	10.85	10.59	9.41	9.41
6	0	19	0	\	1	2	7.68	7.77	7.12	7.12
7	0	8	1	8	0	1	1000	1000	8.28	6.89
8	8	25	0	\	1	1	6.42	11.83	7.22	7.90
9	8	40	1	10	0	1	6.21	9.53	8.63	8.01
10	10	27	0	\	1	1	1000	11	10.65	10.41
11	0	15	0	\	0	3	6.87	6.25	6.45	5.94
						ready time of line	0	4	0	2.5
						initial RoHS status of lines[h]	0	1	1	1

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

Optimal solution to Test-n11k4 obtained by CPLEX:

Integer optimal

Objective = 8.14490000000e-000

Solution time = 100198 sec.

Iterations = 1863477209

Nodes = 86954671

Variable Name	Solution Value	Variable Name	Solution Value
L07	0.550000	L08	0.750000
L04	6.080000	L10	0.490000
Cmax	27.49		
x001101	1.000000	x110501	1.000000
x051201	1.000000	x000302	1.000000
x030102	1.000000	x010902	1.000000
x091202	1.000000	x000703	1.000000
x070603	1.000000	x060803	1.000000
x081203	1.000000	x000204	1.000000
x020404	1.000000	x041004	1.000000
x101204	1.000000		
w1101	0.27	w0501	9.14
w0302	5.00	w0102	8.81
w0902	13.31	w0703	0.27
w0603	11.14	w0803	18.53
w0204	2.77	w0404	7.00
w1004	17.08		

All other variables are zero.

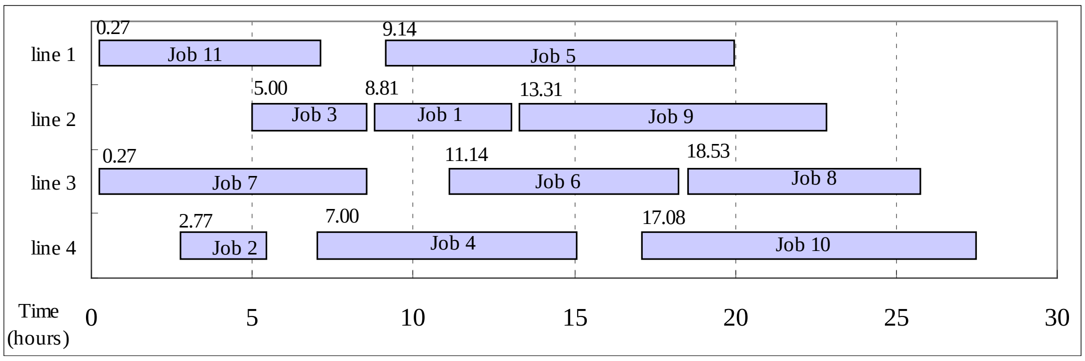


Figure IV-1 A Gantt chart for the optimal schedule of Test-n11k4

APPENDIX V

THE MLSP INSTANCE Test-n12k4 AND ITS SOLUTION

Table V-1 Data for the MLSP instance Test-n12k4

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i [f]	process time[g]			
							line 1	line 2	line 3	line 4
1	0	15	0	\	0	1	1000	5.23	4.68	5.24
2	0	14	0	\	1	1	2.45	3.36	2.45	2.65
3	5	20	1	4	0	2	3.79	4.54	3.57	3.97
4	5	20	0	\	0	1	8.96	8.29	1000	8.08
5	3	15	1	6	1	3	9.85	1000	10.41	10.41
6	3	15	0	\	1	1	7.68	7.77	8.12	8.12
7	0	23	1	8	0	1	1000	1000	8.28	7.89
8	8	22	0	\	1	3	7.42	6.83	1000	7.90
9	5	21	1	10	0	1	9.21	9.53	8.63	8.01
10	5	21	0	\	1	1	1000	11	10.65	10.41
11	0	25	0	\	0	1	6.87	7.25	6.45	1000
12	0	24	0	\	1	1	4.87	5.04	5.65	4.98
						ready time of line	3	0	0	0
						initial RoHS status of lines[h]	0	0	1	0

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

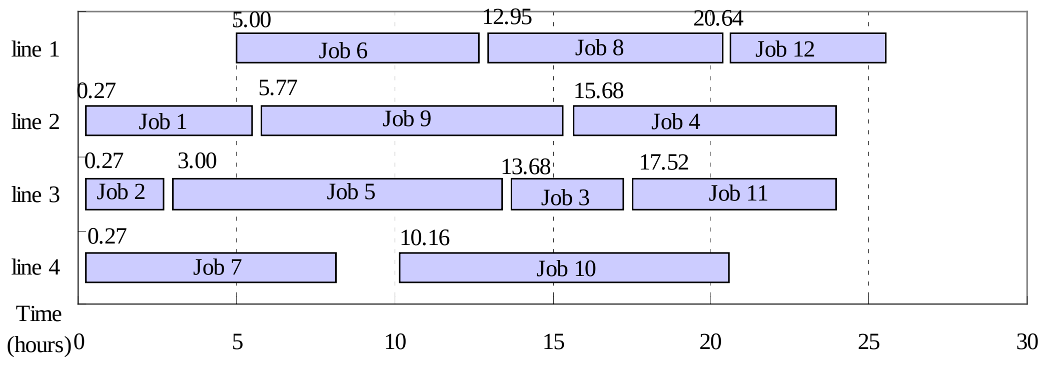


Figure V-1 A Gantt chart for the GA solution of Test-n12k4

APPENDIX VI

THE MLSP INSTANCE Test-n15k4 AND ITS SOLUTION

Table VI-1 Data for the MLSP instance Test-n15k4

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i [f]	process time[g]			
							line 1	line 2	line 3	line 4
1	5	18	0	\	1	1	9.20	9.32	8.14	8.78
2	0	24	1	3	1	2	1000	1000	8.28	7.89
3	0	24	0	\	1	2	7.76	6.44	1000	7.56
4	10	16	0	\	0	1	4.81	5.15	5.54	4.41
5	5	30	0	\	0	3	2.67	3.45	2.57	2.68
6	8	27	1	7	0	3	1000	5.88	4.43	5.07
7	8	27	0	\	0	3	9.08	1000	10.80	10.74
8	5	20	0	\	0	2	7.71	7.34	8.57	8.87
9	6	26	0	\	1	1	6.14	7.33	6.40	1000
10	12	28	0	\	1	1	1000	11.00	10.71	10.44
11	5	26	0	\	0	3	4.68	5.40	5.74	4.87
12	0	30	1	13	0	1	3.98	4.70	3.41	3.89
13	0	30	0	\	0	1	8.76	8.01	1000	8.03
14	2	25	0	\	1	1	1000	11.21	10.87	10.55
15	10	42	0	\	0	1	6.57	7.01	6.70	1000
ready time of line							2.7	1.4	0	0.50
initial RoHS status of lines[h]							1	0	0	1

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

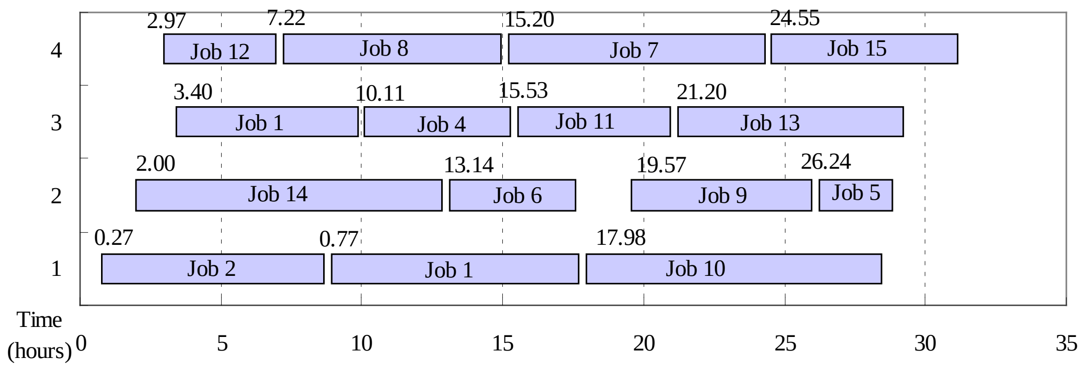


Figure VI-1 A Gantt chart for the GA solution of Test-n15k4

APPENDIX VII

THE MLSP INSTANCE Test-n20k4 AND ITS SOLUTION

Table VII-1 Data for the MLSP instance Test-n20k4

job number	ready time[a]	due date[b]	*if_front*[c]	*bk_job*[d]	*RoHS*[e]	p_i[f]	process time[g]			
							line 1	line 2	line 3	line 4
1	4	22	0	\	1	1	4.81	5.15	5.54	4.41
2	0	18	1	3	1	2	8.76	8.01	1000	8.03
3	0	18	0	\	1	2	1000	11.21	10.87	10.55
4	7	25	0	\	0	1	1000	1000	8.28	7.89
5	8	15	0	\	0	3	7.76	6.44	1000	7.56
6	8	22	1	7	0	3	4.81	5.15	5.54	4.41
7	8	22	0	\	0	3	2.67	3.45	2.57	2.68
8	12	35	0	\	0	2	7.76	6.44	1000	7.56
9	8	35	0	\	1	1	9.08	1000	10.80	10.74
10	8	26	0	\	1	1	7.71	7.34	8.57	8.87
11	0	25	0	\	0	3	6.14	7.33	6.40	1000
12	0	40	1	13	0	1	2.67	3.45	2.57	2.68
13	0	40	0	\	0	1	1000	5.88	4.43	5.07
14	12	33	0	\	1	1	6.57	7.01	6.70	1000
15	10	40	0	\	0	1	9.20	9.32	8.14	8.78
16	5	25	0	\	1	1	3.98	4.70	3.41	3.89
17	5	25	0	\	0	3	1000	11.00	10.71	10.44
18	10	38	1	19	1	3	4.68	5.40	5.74	4.87
19	10	38	0	\	1	3	1000	5.88	4.43	5.07
20	7	20	0	\	1	2	9.08	1000	10.80	10.74
						ready time of line	1.5	0	0.8	2.5
						initial RoHS status of lines[h]	0	0	1	0

[a] ready time for the job

[b] due date for the job

[c] indicator for a front-side job. Value one indicates that the job is a front-side job, zero otherwise

[d] corresponding back-side job number

[e] indicator for RoHS jobs. Value 1 indicates that the job processes RoHS boards, 0 otherwise

[f] weight for tardiness penalty (weight for makespan penalty is set to 0.01)

[g] process time for each job on each line. If a job cannot be processed on the line, the time is set to 1000

[h] initial RoHS status of the line

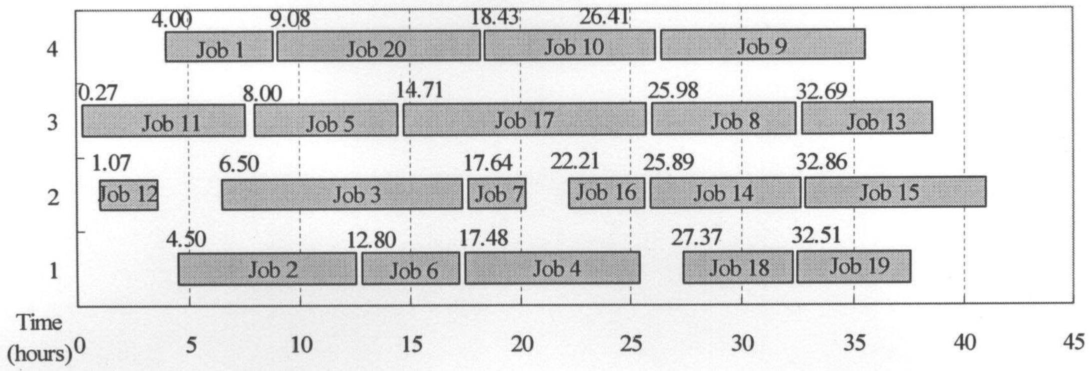


Figure VII-1 A Gantt chart for the GA solution of Test-n20k4